Length-preserving Bit-stream-based JPEG Encryption

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

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

ABSTRACT

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

Categories and Subject Descriptors

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

General Terms

Algorithms, Security, Experimentation

Keywords

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

1. INTRODUCTION

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

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

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

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

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

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

Note that our bit-stream-based approach is designed for en-

MM&Sec'12, September 6-7, 2012, Coventry, United Kingdom.

cryption without the need for recompression, which is useful when there is no possibility to intercept the encoding process. One practical use case is the encryption of pictures from surveillance cameras, most of which send streams of JPEG pictures (which are already encoded). Although real-time recompression is possible with state-of-the-art hardware, omitting this step may save equipment and costs.
This paper is structured as follows: In section, 2 we describe our approach. In section 3, we give an estimation of the effort necessary for a successful attack on a picture encrypted with our approach. Finally, we provide an outlook in section 4 before concluding the paper.

2. BIT STREAM ENCRYPTION

In baseline JPEG, $8 \cdot 8$ blocks of cosine-transformed quantized AC coefficients are zig-zag scanned and run-length coded before being Huffman coded. Hereby, the Huffman code words only encode run-length pairs as symbols, whereas the actual coefficient values are written directly to the bit stream as signed residue from 0 or $-2^s + 1$, respectively, where s denotes the length part of the run-length symbol.
Our approach consists of three different operations. Firstly, the order of the run-length coded symbols together with their corresponding coefficient values (referred to as code-word-value pairs henceforth) is permuted. Secondly, the coefficient value bits are scrambled and thirdly, the order of all blocks within an Interleaved Minimum Coded Unit (iMCU) which use the same Huffman table is permuted. Both, the second and third operation, are described in detail at the end of this section, while the subsequent paragraphs describe the first operation, i.e. the permutation of the order of code-word-value pairs.
Permutations of this order lead to a change of the order of the zero runs in each block, thereby altering the positions of the coefficient values within the $8 \cdot 8$ block. Figure 1 depicts this by example.
On the top left, an examplary block with four non-zero coefficient values is shown. The dots denote that the rest of the coefficients are zero. The DC coefficient is neither changed nor considered. On the top right, the zig-zag scanned values of the examplary block are depicted and grouped with their preceding zeros. Each of these groups is coded as a Huffman code word (black) of the run-length symbol (depicted on top of each Huffman code word) and the coefficient value (grey). E.g., the first coefficient (5), which is preceded by no (i.e. 0) zeros, requires 3 bits (101) to be repesented, thus leading to a run-length of 0/3. As the rest of the blocks' coefficients apart from the four ones depicted are zero, an End of Block (EOB) is signalled.
By swapping the groups of Huffman code words and coefficient values (if there is more than one group), the zero runs and therefore the position of the coefficient values within the block change, as depicted at the bottom of figure 1. However, the bit stream remains format-compliant as the exchange of code words does neither change the Huffman codes themselves nor does it change the total number of coefficients. In addition, it does not change the length of the JPEG file, thus being length-preserving.
The code-word-value pair order permutation through element-wise reordering is derived as follows. Before processing a JPEG file, an Advanced Encryption Standard (AES) [9] implementation in Output Feedback (OFB) mode is initialized with a given initialization vector and key, which can be

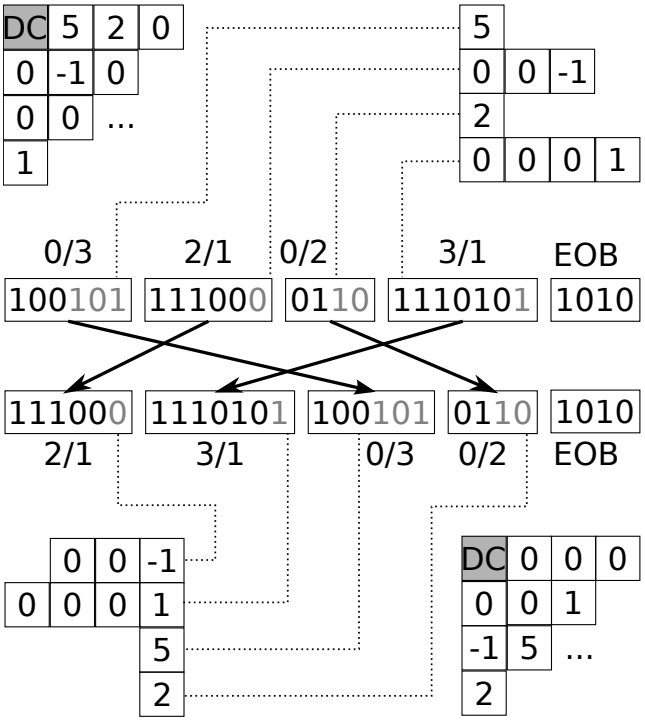

Figure 1: Example of run-length permutation: The order of Huffman coded run-length symbols and their corresponding coefficient values is permuted, thereby changing the position of the values in the coefficient matrix

file- or user-dependent. It then serves as a Pseudo-Random Number Generator (PRNG) by using n AES-encrypted output bits, where 2^n is the desired range of the PRNG. Note that any cryptographic PRNG could be used here.
Code-word-value pair order permutation is then performed by swapping the current code word and its corresponding coefficient value at position i with the code-word-value pair at position $rand(n)$ where $rand$ denotes a call to the AES-based PRNG with an upper value bound of n and n is equal to the number of total code-word-value pairs. For the example in figure 1, $n = 4$, yielding the consumption of 2 encrypted output bits of the AES encoder per possible swap operation.
In addition to code-word-value pair order permutation, our approach changes the coefficients' values in the bit stream (grey bits in figure 1). This is done by toggling each of the n value bits depending on whether or not the AES-based PRNG described above returns a binary zero or one when using one bit. Similar to the run-length order permutation, this does not change the length of the JPEG file as the value bits actually represent a signed residual of fixed size per code word (see above).
Furthermore, the order of all blocks using the same Huffman table within an iMCU is permuted. Figure 2 shows an example with 4:2:0 subsampling [6] where the U and the V block use the same Huffman table (marked grey). After the permutation, the order of U and V is switched, with the bit stream still being format-compliant. The permutation itself is derived as described for run-length permutations above. Note that no code-word-value pairs are exchanged

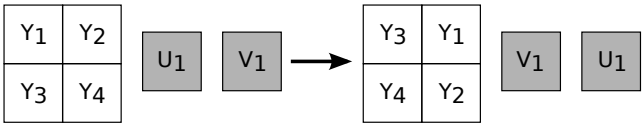

Figure 2: Example of block order permutation: The order of blocks using the same Huffman code words within an iMCU is permuted. In this example, the Y blocks use one set of Huffman code words (white), while both, the U and the V block, use another (grey)

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

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

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

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

3. SECURITY ANALYSIS

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

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

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

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

for all blocks within an iMCU.

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

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

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

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

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

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

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

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

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

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

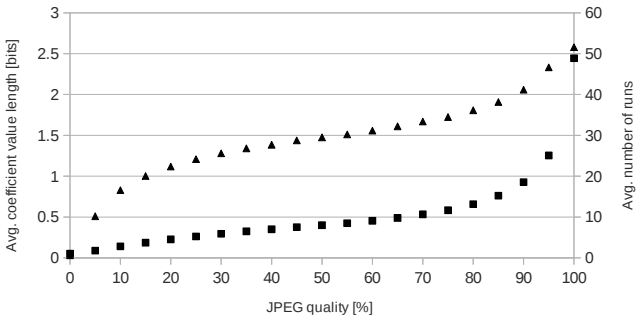

Figure 4: Average number of runs per block (squares) and average coefficient value bit length (triangles) over JPEG quality for the JPEG-compressed LIVE reference picture set [15]

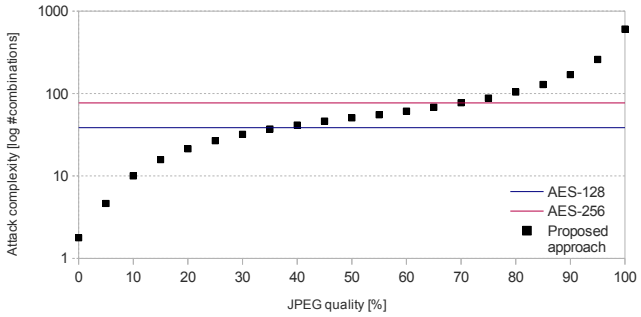

Figure 5: Average attack complexity over JPEG quality for the JPEG-compressed LIVE reference picture set [15] for the proposed approach and AES for comparison

word-value pairs (e.g. 1 or 2) are very unlikely to compromise the content of the whole picture, thus having a low amount of semantic self-information. Note that semantic self-information is hard to measure and therefore requires a justifiable approximation. Thus, we use the average number of code-word-value pairs to represent a block with a medium to high amount of self-information as a practical approximation of a possibly critical block of the picture.
Figure 4 depicts the average number of runs per block and the average length of coefficient values as functions of JPEG quality for the reference pictures of the LIVE data base. Both functions increase monotonically with quality, showing that pictures with finer quantization contain a higher number of runs per block and longer coefficient values. This allows for a higher number of combinations, making an attack on an iMCU harder.
Using the average values $n(q)$ and $l(q)$ at quality q instead of all n_i and $l_{i,j}$, respectively, yields a simplified equation for the overall number $N(q)$ of combinations dependent on the JPEG quality q:

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

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

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

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

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

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

Note that each iMCU can be attacked separately, thereby possibly revealing enough information about the picture that the rest of the picture's iMCUs do not need to be decrypted. This way, the total number of combinations for a full picture does not reprent a valid metric for the number of combinations to try for an attack.
Note that an attacker may eliminate some orderings of code-word-value pairs as high values of high frequency AC coefficients (most of all chroma) are very unlikely to appear in natural images [11]. This reduces the effective values of n_i and $n(q)$, respectively. However, it is hard to quantify the actual reduction as it depends on the picture's content, potentially known signal characteristics and the coefficient distribution of the attacked iMCU's blocks.
Regarding known-plaintext attacks, AES is considered to be not vulnerable [3]. If an attacker has both, the original and the encrypted JPEG picture, deriving the key from the permutations and scrambled bits is nearly as hard as a brute-force attack on the AES key itself [2] which is considered infeasible for 256 bit keys by today's standards.

4. FUTURE WORK

Multiple extensions of our approach are possible and remain future work: firstly, the DC coefficient differences of each block could be scrambled similar to the AC coefficient values, increasing the total number of possible combinations to try for decryption. Secondly, blocks between different iMCUs could be swapped as long as the corresponding blocks in the iMCUs use a the same Huffman tables, yet increasing the total number of possible combinations. Note that both extensions are easy to implement and preserve the length of the bit stream.
Finally, it is possible to use our proposed approach for RoI encryption. iMCUs containing the RoIs can be encrypted while the rest of the picture stays intact. Although limiting the encryption to a set of iMCUs is trivial, signalling them is not, if the length is to be preserved. However, if this limitation is lifted, embedding the RoI information can be done by inserting a comment segment into the bit stream which contains a bitmap of all iMCUs where a one denotes that the iMCU is encrypted, while a zero denotes that it is not. Such a segment increases the file size by the marker size (2 bytes) plus its length field (2 bytes) plus the size of the bitmap, i.e. $\lceil \frac{n_{iMCU}}{8} \rceil$ bytes where n_{iMCU} denotes the number of iMCUs in the picture.

5. CONCLUSION

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

6. ACKNOWLEDGMENTS

This work is supported by FFG Bridge project 832082.

7. REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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