On the Performance of Secret Entropy Coding: A Perspective Beyond Security*

Shujun Li

Abstract In this paper, we study the overall performance of two main forms of secret entropy coding – secret Huffman coding and secret arithmetic coding, as solutions to multimedia encryption. We consider a set of criteria, which include not only security but also other aspects of the performance. We draw the conclusion that neither can fulfill all the criteria, but secret arithmetic coding can offer a better solution. We also point out the possibility of amending existing multimedia coding standards to facilitate multimedia encryption.

1 Introduction

To fulfill increasing demands for content protection of multimedia products, multimedia encryption has been extensively studied in the past [9, 10, 38, 44]. Because simply encrypting compressed multimedia data with a textual cipher cannot fulfill requirements of some practical applications, many different encryption techniques have been proposed to design joint compression-encryption schemes. Among all the proposed techniques, secret entropy coding has attracted more attention than others, because there are various ways to integrate encryption into data compression without much additional computational load. In this context Huffman coding and arithmetic coding are the two most widely-adopted entropy coding algorithms in multimedia coding standards, and most research is done on secret Huffman coding and secret arithmetic coding. Though some cryptanalytic results on a number

Shujun Li
Department of Computer and Information Science, University of Konstanz, Universitätsstraße 10, Mailbox 697, Germany, e-mail: http://www.hooklee.com

* This is the author's preprint of a paper published in *Autonomous Systems: Developments and Trends*, edited by Herwig Unger, Kyandoghere Kyamaky and Janusz Kacprzyk, *Studies in Computational Intelligence*, vol. 391, pp. 389-401, Springer-Verlag GmbH, 2011. The original publication is available at www.springerlink.com.

of specific secret entropy coding algorithms have been reported and some security problems have been identified, many aspects of the overall performance of secret entropy coding have not been well understood and a performance comparison between secret Huffman coding and secret arithmetic coding has never been done before.

This paper studies the overall performance of secret Huffman coding and secret arithmetic coding, by considering a set of criteria. Instead of analyzing security only, we extend our focus to the capabilities to support more useful features required in multimedia encryption. As a conclusion, we point out that neither secret Huffman coding and secret arithmetic coding can fulfill all criteria, but the latter can offer a better solution to multimedia encryption as a whole.

The rest of the paper is organized as follows. In the next section, we give a brief survey of multimedia encryption and show a set of criteria about the overall performance of multimedia encryption systems. Then, we apply these criteria to secret Huffman coding and secret arithmetic coding in Section 3, trying to clarify their overall performance and make a qualitative comparison between them. Finally in the last section, we give a short summary and mention the possibility of designing security-oriented multimedia coding systems to facilitate multimedia encryption.

2 Multimedia Encryption

According to the relationship between encryption and compression, there are three possible approaches to design multimedia encryption systems: 1) encryption before compression; 2) encryption after compression; 3) joint compression-encryption.

For the first approach, completely new algorithms have to be devised to ensure efficient compression of encrypted data, because encryption generally leads to a random output with a very high information entropy that cannot be compressed effectively afterwards. A recent solution based on distributed source coding was proposed in [20], by taking the encryption key as side information available at the decoder side. However, for this solution, decoding is generally impossible without the knowledge of the decryption key, which is not desired in some applications requiring format compliance, such as postprocessing without decryption (see below for more details). In addition, this scheme puts some requirements on the encryption algorithm involved such that not all available ciphers can be freely chosen and deployed. Another problem is that the proposed solution is not compatible with existing multimedia coding standards.

The second approach is the most direct and simplest one, which is often called *naive encryption* in the literature [32]. By simply employing a textual cipher in this way, the syntax format of the compressed multimedia data will be destroyed. However, *format compliance* of encrypted multimedia data is very useful in many applications such as the following:

- *postprocessing of encrypted multimedia data without decryption*: watermark embedding, transcoding, rescrambling, bitrate control, repacketization, etc.;

- *perceptual/traparent encryption* [24]: encryption is used to degrade the quality of multimedia data rather than conceal all the information, which is useful for preview-before-pay multimedia services;
- *scalable encryption*: a multimedia product has different resolutions encrypted with different configurations;
- *ROI (region-of-interest) encryption*: only part of a multimedia product is encrypted.

To achieve format compliance, it is obvious that the multimedia data cannot be fully encrypted, i.e., the idea of *selective encryption* (also called *partial encryption*) has to be adopted to leave some syntax elements unencrypted. In addition, special aspects have to be considered in the design of the encryption part such that the encryption process is compatible with the underlying multimedia coding standard. This means that the third approach – joint compression-encryption – should be used instead of the first two ones.

Selective encryption is also very useful to reduce the encryption load, which is especially important for some applications such as video-on-demand systems that need to perform real-time encryption on a large number of videos and send the encrypted videos to a large number of users simultaneously. Selective encryption is also useful to save energy for resource-constrained devices like wireless multimedia sensor networks (WMSNs) [2]. Unfortunately, if too many syntax elements containing perceptual information are left unencrypted, the security might be compromised. A lot of research [1, 8, 23, 24, 26, 31, 37, 39, 41] has shown that some perceptual information can be recovered from various kinds of unencrypted syntax elements. This problem is due to the following fact: most multimedia coding standards are designed in such a way that many syntax elements can be decoded independently without decoding other syntax elements. To essentially overcome (or at least mitigate) this problem, the underlying multimedia coding standards have to be amended by introducing more dependence among syntax elements containing aural/visual information. But long-range dependence should be avoided, otherwise ROI encryption will be impossible.

Because many joint compression-encryption systems achieve encryption at the expense of compression efficiency, the size of ciphertext will not be the same as that of the plaintext. As a result, *size preservation* becomes another important concern in the design of multimedia encryption systems. In the ideal case, every syntax element should keep its original size after encryption. Size preservation is a stronger version of "no influence on compression efficiency". Typical applications of ideal size preservation include on-the-spot encryption[2], real-time mounting and dropping of encryption, simultaneous encryption at multiple points, and so on. In addition, if size preservation is fulfilled, bitrate re-control and re-packtization will not be necessary after encryption.

Yet another concern is about the capability to support the reuse of session key. Because of the nature of some multimedia encryption techniques, block ciphers can-

[2] On-the-spot encryption means that a file can be encrypted by simply writing the ciphertext back to the original place of the plaintext without creating a temporary copy.

not be used. In other words, only pseudo-random keystreams generated from session keys (i.e., stream ciphers) can be used. Because stream ciphers are not secure when any session key is reused, a key management system must be employed to determine a unique session key for each encrypted multimedia signal, which will make the whole system overcomplicated for some practical applications such as encryption of private pictures and videos on personal computers and hand-held devices.

Because there are many concerns of a multimedia encryption system, it is not sufficient to evaluate the overall performance with a single factor like security. We believe at least the following criteria (or requirements) should be considered:

1. security against various attacks;
2. ideal format compliance;
3. ideal size preservation;
4. reuse of session key (or flexibility – capability to work with both stream ciphers and block ciphers);
5. low encryption load (vs. naive encryption);
6. high energy efficiency (i.e., low energy consumption);
7. easy implementation in existing multimedia coders.

To our knowledge, the existing work has not considered the overall performance of multimedia encryption systems with respect to all of the above criteria, though some results have been reported on security, compression efficiency (weaker version of size preservation) and some implementation issues.

In the next section, we apply the above criteria to secret Huffman coding and secret arithmetic coding, trying to obtain a better understanding on the overall performance of the two main forms of secret entropy coding.

3 Secret Entropy Coding

First, we show some common issues about secret entropy coding, which are about the last three criteria.

Secret entropy coders realize encryption by keeping the statistical models and/or the behaviors of the entropy coders secret. Because an entropy coder has already been embedded in each multimedia coder, it is generally very easy to integrate encryption into the whole multimedia coding process. This is the main advantage of secret entropy coding, and also the main reason why it has attracted more attention than other multimedia encryption techniques.

There are two kinds of secret entropy coders: static and dynamic. A static secret entropy coder has a static statistical model and static coding behavior. It can be easily converted into a dynamic coder, by using a pseudo-random source (i.e., a stream cipher) to frequently update the statistical model and/or the coding behavior. In contrast, there does not seem to be a way to directly use block ciphers. At least no work has been reported on such a possibility. As a result, secret entropy coding both suffer from a common problem: the session key cannot be reused.

For static secret entropy coders, no encryption load is added to the base multimedia coding system, because there are no explicit encryption operations except for the initial process of generating secret statistical models and/or coding behaviors. This is another major advantage of static secret entropy coding. For dynamic secret entropy coders, the condition is quite different. Since a stream cipher has to be used to frequently update the statistical model and/or the coding behavior, there is some additional encryption load. Because the update is carried out before compression, the encryption load consumed on the update will be more than that consumed on naive encryption as long as the update frequency f is larger than a critical value f_0. Assuming the computational complexity of the updating process is n times more complicated than that of the stream cipher involved, then the critical frequency will be $f_0 = r/n$, where $r = \mathrm{Size(input)}/\mathrm{Size(output)}$ is the compression ratio.

3.1 Secret Huffman Coding

Huffman coding is the most widely-adopted entropy coding algorithm in multimedia coding standards, such as JPEG [15], MPEG [14, 16, 17], H.264/AVC [18] and VC-1 [33]. Given a prior statistical model of the input, a Huffman tree is designed by assigning bit patterns of different sizes (i.e., different variable-length codewords – VLCs) to different nodes (i.e., different input symbols). For an input sequence of symbols, the output of a Huffman encoder is a sequence of VLCs. The Huffman tree is constructed in such a way that no VLC is the prefix of any other VLCs and thus a bitstream of VLCs can be decoded unambiguously. By assigning longer VLCs to input symbols with smaller occurrence probabilities, the effect of data compression is thus achieved. In multimedia coding standards, each Huffman tree involved are normally represented by a 1-D Huffman table, so the Huffman coding process can be performed via simple look-up-table operations. In newer multimedia coding standards like H.264/AVC, context-adaptive Huffman coding (formally named CAVLC – context-adaptive variable-length coding) is used. In this case, there are a number of candidate Huffman tables for each input symbol, and which one will be used is determined by previously-coded values (i.e., the context).

As we mentioned above, to design a secret entropy coder, one needs to keep the statistical models and/or the coding behavior secret. For secret Huffman coding, this means keeping one or more secret Huffman tables secret.

The most serious problem about secret Huffman coding is its incapability to maintain format compliance. There are three kinds of format incompliance. First, for any secret Huffman table, once there is some VLCs that are neither valid VLCs nor prefixes of valid VLCs in the original Huffman table, it will be impossible to decode these VLCs. Second, if some VLCs in a secret Huffman table are not included in the original table but are prefixes of some valid VLCs, the synchronization between encoder and decoder might be destroyed and as a result the decoding of the whole bitstream will fail at some point after any VLC is incorrectly decoded. Third, even when all the VLCs in a secret Huffman table are also valid VLCs in the original table, i.e., the secret table is obtained by permuting the original one, some semantic errors might still happen during the decoding process. For instance, the number of DCT coefficients in a single block might exceed the theoretical upper bound (64 for an 8×8 block), since the number of zero coefficients is encoded in each VLC. Actually, the above results are common for all entropy coding algorithms in which VLCs are involved, such as Exp-Golomb coding used in H.264/AVC.

Though the integration of encryption into multimedia coding is also quite simple for secret Huffman coding, there is a potential problem about implementation. In some multimedia coding systems, the source code of the entropy coder is specially optimized for the original Huffman tables. So re-programming and re-compilation of the entropy coder might be necessary, though it is not heavy work in most cases.

In the following, we discuss the other two criteria – size preservation and security – for static and dynamic Huffman tables, respectively.

3.1.1 Static Huffman Tables

In most multimedia coding standards, especially those relatively old ones, static Huffman tables are used. Secret Huffman coding algorithm can be easily designed by replacing these static Huffman tables with secret (but also static) ones. Instead of designing secret Huffman tables from scratch, many researchers have suggested deriving them from the original ones defined in multimedia coding standards by performing some specific operations such as the following ones: 1) permuting VLCs in the original Huffman table [6,21]; 2) tree mutation process – randomly swapping the bit patterns assigned to two branches at the same level of the Huffman tree [41]; 3) randomly flipping the last bits of some VLCs and adjusting other VLCs accordingly to ensure the validity (i.e., the prefix-rule) of the Huffman tree [21]. A stream cipher is generally used as a pseudo-random source to control the operations involved.

For static Huffman tables, there exists a conflict between size preservation and security. To achieve size preservation, the secret Huffman tables should have the same structure as the original ones, and thus each VLC has the same size as designed in [6, Algorithm I]. Unfortunately, as long as the size of each VLC does not change, the secret Huffman tables can be easily revealed in known/chosen plaintext attack, because the boundary between any two consecutive VLCs is obviously distinguishable. To solve this problem, some researchers proposed to relax the requirement on size preservation and allow some VLCs to have different sizes from the original ones [21,41]. Unfortunately, such a relax of size preservation does not enhance security, because in plaintext attacks the size of each VLC can still be recognized by observing the context of the encrypted bitstream [19,25]. For example, when $n \geq 2$ identical VLCs occur consecutively, it is quite easy to locate the repeated bit pattern.

Another security problem is about the number of "good" secret Huffman tables. Because of the nature of entropy coding, different input symbols have different occurrence probabilities, and thus different VLCs in a Huffman table have different levels of significance – shorter VLCs are more significant. However, the number of significant VLCs is often much smaller than the number of other less significant VLCs. This means that the number of “good” secret Huffman tables is mainly determined by a small number of significant VLCs. This fact can explain the experimental results about MPEG-2 videos reported in [21]: a large number of candidates exist for each secret Huffman table, but only a quite small number of them are “good” for encryption. An implication of this fact is that the attacker does not need to break all VLCs. Instead, a partially-broken Huffman table might be enough to recover most information of a multimedia signal encrypted by a secret Huffman table.

To enlarge the relatively small key space of a single Huffman table, one can try to increase the number of secret Huffman tables, which is possible because normally more than one Huffman table is defined in each multimedia coding standard. For example, in MPEG-2 standard, 15 Huffman tables are defined and 5 of them are used in the secret Huffman coding algorithm proposed in [21]. Unfortunately, because these distinct Huffman tables are defined for coding different syntax elements, a divide-and-conquer (DAC) attack might be mounted to break all the secret Huffman tables separately. For instance, for the secret Huffman coding algorithm proposed in [21], the secret Huffman table B-14 can be separately broken by trying to decode a number of non-intra macroblocks, while other Huffman tables are still unknown. As a whole, static secret Huffman tables cannot offer a high level of security.

3.1.2 Dynamic Huffman Tables

To improve the security of static Huffman tables, dynamic Huffman tables can be used instead of static ones. There are three approaches to generate dynamic Huffman tables.

The first approach is so-called MHT (multiple Huffman tables) encryption [41]. That is, for each input symbol to be encoded and encrypted, a Huffman table is secretly selected from a number of (maybe public) candidate Huffman tables. In this case, a stream cipher should be used as a pseudo-random source to choose a specific Huffman table for each VLC. If the candidate Huffman tables are public and the session key is reused for distinct multimedia signals, it has been shown [19] that known/chosen plaintext attack can be launched to recover all the dynamic Huffman tables one by one. If the candidate Huffman tables are also kept secret, differential chosen-plaintext attack can be used to recognize VLCs in each secret Huffman table by observing the change of ciphertexts when only one VLC at a given position changes. So the security can be ensured only when the session key is not reused. Note that other problems of static Huffman tables cannot be overcome.

The second approach is to dynamically update the secret Huffman table after every n VLCs have been coded. This approach can be considered as a special case of the first one, where all the candidate Huffman tables are kept secret. Apparently, the analysis on the first approach remains the same for the second one.

The third approach is to use context-adaptive Huffman tables. In this case, dynamic Huffman tables are determined by previously-coded symbols (i.e., the context). Apparently, it is also a special case of the first approach.

3.2 Secret Arithmetic Coding

Arithmetic coding is a quite different entropy coding algorithm from Huffman coding, which represents each input symbol s as a subinterval $I(s)$ in the range $[0,1)$ rather than a VLC in Huffman coding. The length of the subinterval is determined by the occurrence probability of the corresponding input symbol. The encoding process starts from the unit interval $I_0 = [0,1)$, and a smaller subinterval I_i is obtained once a new symbol s_i is fed in, where the position and length of I_i is determined by $I(s_i)$, i.e., by the occurrence probability of s_i. After all n symbols have been processed, any fraction in the final subinterval I_n can be taken as the output of the encoder. Generally speaking, arithmetic coding is an optimal entropy coding and has a better compression performance than Huffman coding. In addition, it is more natural to introduce context adaptiveness in arithmetic coding than in Huffman coding, thanks to the separation of the statistical model and the coding process. In fact, most arithmetic coders available are context-adaptive.

Though arithmetic coding has not been adopted in multimedia coding standards until recently, the possibility of adding encryption into compression has attracted much attention since the very beginning of the development of arithmetic coding technique. After the appearance of the first proposal in 1988 [40], a lot of follow-up research have been reported [3–5, 7, 11–13, 19, 22, 27–30, 34–36, 41, 43]. Basically, there are three classes of secret arithmetic coding algorithms:

- *secret initial statistical model* – the initial statistical model is taken as the key, and the coder works as usual;
- *secret initialization process* – the initial statistical model is public, and a secret initialization process is carried out before encoding/decoding starts;
- *randomized coding* – a stream cipher is used to randomize the arithmetic coding process.

It has been known that the first two classes are not secure against chosen-plaintext attacks [4, 5, 27, 35]. A combination of the two classes of secret arithmetic coding algorithms is also insecure against chosen-plaintext attack, as shown in [34]. The third class of secret arithmetic coders are secure, but only when the session key is not reused [19, Sec. IV.A].

To improved the security of the first two classes of secret arithmetic coding algorithms, some amendments have been suggested [27, 35, 36]. One amendment is masking the output of the encoder with a secret pseudo-random keystream, which indeed can improve the security, but is just an additional level of security, not an essential enhancement on the security of the original secret arithmetic coding scheme. Another amendment is frequently resetting the statistical model, which improves the security at the expense of compression efficiency. Other amendments also have various weaknesses and limitations.

Besides the above security problems, there are also problems about size preservation and format compliance. As long as the statistical model is changed, a secret arithmetic coder will be unable to achieve the same compression efficiency as the original one, so size preservation cannot be maintained. Similarly, because different statistical models lead to compressed data of different sizes, the synchronization between encoder and decoder will be destroyed, when an encrypted multimedia signal is decoded by an decoder without the knowledge of the key. That is, format compliance cannot be maintained, either.

According to the above analysis, to maintain size preservation and format compliance, the statistical model should remain untouched. In other words, only the behavior of the arithmetic coder can be secretly modified. Some secret arithmetic coding algorithms were proposed following this idea [11, 13], which change the position of the subinterval corresponding to each input symbol. Because the length of each subinterval of each input symbol remains, the compression efficiency will not be influenced in principle. As a result, it might be possible to maintain format compliance and size preservation simultaneously. Experiments in [11] have shown that such an ideal result can be achieved for JPEG2000 standard. But the condition is different for H.264/AVC standard, in which the arithmetic coder is often used to code VLCs and the termination of the coding process is related to the current coded VLC itself. In this case, any decoding error will definitely lead to the loss of synchronization between encoder and decoder, and thus format incompliance happens.

4 Conclusion

Table 1 Performance comparison of secret Huffman coding and secret arithmetic coding.

	Huffman coding	arithmetic coding
security	yes (session key not reused)/no	
format compliance	no	yes/no
size preservation	no	yes/no
reuse of key	no	
encryption load	very low/conditional	
implementation	easy	

Table 1 shows a summary of the results obtained in the last section, from which we can see that neither secret Huffman coding nor secret arithmetic coding can fulfill all the criteria, but secret arithmetic coding can offer a better solution as a whole. The main advantage of secret arithmetic coding is its potential to maintain size preservation and format compliance for some multimedia coding standards.

Considering the compression efficiency of arithmetic coding is also better, we propose to use the arithmetic coding in any new multimedia coding standards.

Recalling the reason why some criteria cannot be fulfilled by secret Huffman coding and secret arithmetic coding, we see possibilities to change the status by making some security-oriented amendments to existing multimedia coding standards, such as the following ones:

- *adding a compulsory error-tolerance mechanism*, which can help relax the requirement on format compliance and allow some kind of minor decoding errors (for example, the loss of synchronization when decoding a macroblock);
- *adding content-dependent IDs*, which can be employed by stream ciphers as initial vectors to generate different session keys for different plaintexts;
- *limiting the use of VLCs*, which will help alleviate the side effect of VLCs on format compliance;
- *introducing termination markers (like those used in JPEG2000 standard)*, which might help to maintain format compliance for secret arithmetic coders when VLCs are encoded.

In addition, to overcome the conflict between security and selective encryption, more dependence among syntax elements within a small area might be added or enhanced.

Note that some simple methods might be able to fulfill all performance criteria if some amendments are made to existing multimedia coding standard. For instance, if there is an error-tolerance mechanism and context-adaptive entropy coding (either Huffman coding or arithmetic coding) is used, one can simply selectively encrypt the n leading bits of the entropy encoder's output. As shown in [42], such a selective encryption might be able to achieve an effect of "virtual full encryption" when some conditions are satisfied. We can see that all other criteria can be fulfilled easily, too.

To sum up, considering the fact that an ideal solution to multimedia encryption cannot be easily found for most existing multimedia coding standards, we do believe that new security-oriented standards should be developed. In future we will focus our research on this direction and try to show the feasibility to work out such new standards based on existing ones.

5 Acknowledgments

Shujun Li was supported by a fellowship from the Zukunftskolleg ("Future College") of the University of Konstanz, Germany, which is part of the "Excellence Initiative" Program of the DFG (German Research Foundation).

References

1. Agi, I., Gong, L.: An empirical study of secure MPEG video transmission. In: Proc. ISOC Symposium on Network and Distributed Systems Security (SNDSS'96), pp. 137–144 (1996)
2. Akyildiz, I.F., Melodia, T., Chowdhury, K.R.: Wireless multimedia sensor networks: Applications and testbeds. Proc. IEEE **96**(10), 1588–1605 (2008)
3. Barbir, A.: A methodology for performing secure data compression. In: Proc. Twenty-Ninth Southeastern Symposium on System Theory (SSST'97), pp. 266–270. IEEE (1997)
4. Bergen, H.A., Hogan, J.M.: Data security in a fixed-model arithmetic coding compression algorithm. Computers & Security **11**(5), 445–461 (1992)
5. Bergen, H.A., Hogan, J.M.: A chosen plaintext attack on an adaptive arithmetic coding compression algorithm. Computers & Security **12**(2), 157–l67 (1993)
6. Bhargava, B., Shi, C., Wang, S.Y.: MPEG video encryption algorithms. Multimedia Tools and Applications **24**(1), 57–79 (2004)
7. Bose, R., Pathak, S.: A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system. IEEE Trans. Circuits and Systems–I **53**(4), 848–857 (2006)
8. Engel, D., Stütz, T., Uhl, A.: Format-compliant JPEG2000 encryption in JPSEC: Security, applicability and the impact of compression parameters. EURASIP J. Information Security **2007**, art. no. 94,565 (2007)
9. Furht, B., Kirovski, D. (eds.): Multimedia Security Handbook. CRC Press, LLC (2004)
10. Furht, B., Muharemagic, E., Socek, D. (eds.): Multimedia Encryption and Watermarking. Springer (2005)
11. Grangetto, M., Magli, E., Olmo, G.: Multimedia selective encryption by means of randomized arithmetic coding. IEEE Trans. Multimedia **8**(5), 905–917 (2006)
12. Irvine, S., Cleary, J., Rinsma-Melchert, I.: The subset sum problem and arithmetic coding. Research Report 95/7, Department of Computer Science, University of Waikato (1995)
13. Ishibashi, H., Tanaka, K.: Data encryption scheme with extended arithmetic coding. In: Mathematics of Data/Image Coding, Compression, and Encryption IV, with Applications, *Proc. SPIE*, vol. 4475, pp. 222–233 (2001)
14. ISO/IEC: Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video. ISO/IEC 11172-2 (MPEG-1) (1993)
15. ISO/IEC: Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines. ISO/IEC 10918-1 (JPEG) (1994)
16. ISO/IEC: Information technology – Generic coding of moving pictures and associated audio information: Video. ISO/IEC 13818-2 (MPEG2), last revised in 2000 (1996)
17. ISO/IEC: Information technology – Coding of audio-visual objects – Part 2: Visual. ISO/IEC 14496-2 (MPEG-4), last revised in 2004 (2001)
18. ITU-T: Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding. ISO/IEC 14496-10, last revised in 2005 (2004). Also published as ITU-T Rec. H.264 in 2003 under the title "Advanced video coding for generic audiovisual services", last revised in 2005
19. Jakimoski, G., Subbalakshmi, K.P.: Cryptanalysis of some encryption schemes for multimedia. IEEE Trans. Multimedia **10**(3), 330–338 (2008)
20. Johnson, M., Ishwar, P., Prabhakaran, V., Schonberg, D., Ramchandran, K.: On compressing encrypted data. IEEE Trans. Singal Processing **52**(10), 2992–3006 (2004)
21. Kankanhalli, M.S., Guan, T.T.: Compressed-domain scrambler/descrambler for digital video. IEEE Trans. Consumer Eletronics **48**(2), 356–365 (2002)
22. Kim, H., Wen, J., Villasenor, J.D.: Secure arithmetic coding. IEEE Trans. Singal Processing **55**(5), 2263–2272 (2007)
23. Li, S., Ahmad, J.J., Saupe, D., Kuo, C.C.J.: An improved DC recovery method from AC coefficients of DCT-transformed images. In: Proceedings of 17th IEEE International Conference on Image Processing (ICIP 2010), pp. 2085–2088 (2010). URL `http://www.hooklee.com/default.asp?t=AC2DC`

24. Li, S., Chen, G., Cheung, A., Bhargava, B., Lo, K.T.: On the design of perceptual MPEG-video encryption algorithms. IEEE Trans. Circuits and Systems for Video Technology **17**(2), 214–223 (2007)
25. Li, S., Chen, G., Cheung, A., Lo, K.T., Kankanhalli, M.: On the security of an MPEG-video encryption scheme based on secret Huffman tables. In: Advances in Image and Video Technology: Third Pacific Rim Symposium, PSIVT 2009, Tokyo, Japan, January 13-16, 2009. Proceedings, *Lecture Notes in Computer Science*, vol. 5414, pp. 898–909. Springer (2009)
26. Li, S., Karrenbauer, A., Saupe, D., Kuo, C.C.J.: Recovering missing coefficients in DCT-transformed images. In: Proceedings of 18th IEEE International Conference on Image Processing (ICIP 2011). IEEE (2011). URL `http://www.hooklee.com/default.asp?t=ICIP2011`
27. Lim, J., Boyd, C., Dawson, E.: Cryptanalysis of adaptive arithmetic coding encryption schemes. In: Proc. Second Australasian Conference on Information Security and Privacy (ACISP'97), *Lecture Notes in Computer Science*, vol. 1270, pp. 216–227. Springer (1997)
28. Liu, X., Farrell, P., Boyd, C.: A unified code. In: Proc. 7th IMA International Conference on Cryptography and Coding, *Lecture Notes in Computer Science*, vol. 1746, pp. 84–93. Springer (1999)
29. Liu, X., Farrell, P.G., Boyd, C.: Resisting the Bergen-Hogan attack on adaptive arithmetic coding. In: Proc. 6th IMA International Conference on Cryptography and Coding, *Lecture Notes in Computer Science*, vol. 1355, pp. 199–208. Springer (1997)
30. Liu, X., Farrell, P.G., Boyd, C.A.: Arithmetic coding and data integrity. In: Proc. Workshop on Coding and Cryptography (WCC'99), pp. 291–299 (1999)
31. Lookabaugh, T.D., Sicker, D.C., Keaton, D.M., Guo, W.Y., Vedula, I.: Security analysis of selectively encrypted MPEG-2 streams. In: Multimedia Systems and Applications VI, *Proc. SPIE*, vol. 5241, pp. 10–21 (2003)
32. Qiao, L., Nahrsted, K.: Comparison of MPEG encryption algorithms. Computers & Graphics **22**(4), 437–448 (1998)
33. SMPTE (Society of Motion Picture and Television Engineers): Standard for television – VC-1 compressed video bitstream format and decoding process. SMPTE 421M (2006)
34. Uehara, T., Safavi-Naini, R.: Attack on Liu/Farrell/Boyd arithmetic coding encryption scheme. In: Proc. IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security (CMS'99), pp. 273–290 (1999)
35. Uehara, T., Safavi-Naini, R.: Attacking and mending arithmetic coding encryption schemes. In: Proc. Australasian Computer Science Conference, pp. 408–419 (1999)
36. Uehara, T., Safavi-Naini, R.: Designing secure arithmetic coding encryption schemes. In: Proc. 22nd Symposium on Information Theory and its Applications (SITA'99), vol. 2, pp. 773–776 (1999)
37. Uehara, T., Safavi-Naini, R., Ogunbona, P.: Recovering dc coefficients in block-based dct. IEEE Trans. Image Processing **15**(11), 3592–3596 (2006)
38. Uhl, A., Pommer, A.: Image and Video Encryption: From Digital Rights Management to Secured Personal Communication. Springer (2005)
39. Wen, J., Severa, M., Zeng, W., Luttrell, M.H., Jin, W.: A format-compliant configurable encryption framework for access control of video. IEEE Trans. Circuits and Systems for Video Technology **12**(6), 545–557 (2002)
40. Witten, I.H., Cleary, J.G.: On the privacy afforded by adaptive text compression. Computers & Security **7**(4), 397–408 (1988)
41. Wu, C.P., Kuo, C.C.J.: Design of integrated multimedia compression and encryption systems. IEEE Transactions on Multimedia **7**(5), 828–839 (2005)
42. Wu, X., Moo, P.W.: Joint image/video compression and encryption via high-order conditional entropy coding of wavelet coefficients. In: Proc. IEEE Conference on Multimedia Computing and Systems (CMS'99), pp. 908–912 (1999)
43. Xie, D., Kuo, C.C.J.: Efficient multimedia data encryption based on flexible QM coder. In: Security, Steganography, and Watermarking of Multimedia Contents VI, *Proc. SPIE*, vol. 5306, pp. 696–704 (2004)

44. Zeng, W., Yu, H., Lin, C.Y. (eds.): Multimedia Security Technologies for Digital Rights Management. Academic Press (2006)