A Key Embedded Video Codec for Secure Video Multicast*

Hao Yin[1], Chuang Lin[1], Feng Qiu[1], Xiaowen Chu[2], and Geyong Min[3]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P.R.China
{hyin, clin, fqiu}@csnet1.cs.tsinghua.edu.cn
[2] Computer Science Department, HongKong Baptist University, Kowloon Tong, Kowloon, Hong Kong
chxw@comp.hkbu.edu.hk
[3] Department of Computing, University of Bradford, Bradford, BD7 1DP, U.K.
g.min@brad.ac.uk

Abstract. The problem of controlling access to multimedia multicasts requires the distribution and maintenance of keying information. This paper proposes a new key embedded video codec for the media-dependent approach that involves the use of a data-dependent channel, and can be achieved for multimedia by using data embedding techniques. Using data embedding to convey keying information can provide an additional security and reduce the demand of bandwidth. In this paper, we develop a new statistical data embedding algorithm on compression-domain, then, by combining FEC (Forward Error Correction) algorithm and our proposed key information frame format. After a series of simulation experiments, we find that the new codec can satisfy the special demand of media-dependent approach. At the same time, the codec provides good performance.

1 Introduction

Access control in video multicast is usually achieved by encrypting the content using an encryption key, known as the session key (SK) that is shared by all legitimate group members [1] [2]. Since the group membership will most likely be dynamic with users joining and leaving the services, it is necessary to change the SK in order to prevent the leaving user from accessing future communication and prevent the joining user from accessing prior communication. In order to update the SK, a party responsible for distributing the keys, called the group center (GC), must securely communicate the new key material to the valid users. This task is achieved by transmitting rekeying messages that use key encrypting keys (KEKs) to encrypt and distribute new keys. In addition, any solution to access control should address issues of resource scalability for scenarios of large privileged groups.

* This work is supported by the National Natural Science Foundation of China (No. 60372019, 60429202, and 60473086), the Projects of Development Plan of the State Key Fundamental Research (No. 2003CB314804).

J. Dittmann, S. Katzenbeisser, and A. Uhl (Eds.): CMS 2005, LNCS 3677, pp. 77 – 87, 2005.

The problem of access control for multicasts has been recent attention in the literature, and several efficient schemes have been proposed with desirable communication properties. These schemes can be classified into two distinct classes of mechanisms: media-independent mechanism and media-dependent mechanism [1]. In media-independent mechanism, the rekeying messages are conveyed by a means totally disjoint from the multimedia content; while in media-dependent mechanism, the rekeying messages are embedded in the multimedia content and distributed to those who receive the data.

When media-dependent mechanism uses data embedding techniques to delivery rekeying messages, the issues of reliability of them and transparency for adaptation mechanisms become more pronounced than in the media-independent case. New video codec should face these challenges: 1) transparency for adaptation mechanism;
2) reliability for rekeying message delivery; 3)real-time key embedding;4) Multicast architecture and packetization independence;5) Video Quality; since the rekeying messages are embedded into the video content by modifying the original signal, there will be great impact for video quality. How to make the video quality after key embedding acceptable for users and meanwhile maintain preferable security is another challenge.

In this paper we propose a new key embedded video codec which can solve the challenges mentioned in above. The rest of the paper is organized as follows Section 2 presents our key embedded video codec, including the error resilient embedded video coding and detecting scheme. Section 3 evaluates our codec by a series of simulation experiments. Section 4 contains some concluding remarks. Appendix A proves the validity of our key embedding scheme. Appendix B presents the relationship between luminance in spatial domain and the coefficients in the DCT domain, and infers the luminance alteration method in DCT domain.

2 Error Resilient Embedded Video Coding Scheme

2.1 Overview

The whole the real-time key embedding and detecting process is divided into two parts: key embedded video coding part and key detecting & decoding part. In the key embedded video coding part, key embedding algorithm is added into the process of source coding of MPEG2 video. In this algorithm, key message will be modulated into the DCT coefficients. In order to improve the error resilience of the embedded key messages, we make use of FEC approach to conduct error control. Then, all messages will be encrypted by old key and sent out to another host via the network. In key detecting & decoding part, firstly we use the shared key to decrypt the incoming data packet, and decoding them. While decoding it, we detect the embedded key messages, which will be the shared key at the next communication interval and be used to decrypt the incoming packets.

2.2 Key Embeding and Key Detecting Algorithm

In this section, we use the ALV (Average Luminance Value) modulation in the DCT domain to implement this algorithm. When the MPEG2 bitstream arrives to the

transcoder, the transcoding algorithm can be abstracted to perform in a way that first decodes the MPEG2 bitstream, then down-samples the YUV sequence and re-encodes the bitstream into MPEG4 or H.264 bitstream. It is the re-encoding process, specifically, the re-quantization process that makes the video signal different from the original content. As for the ALV of a field, since it is averaged by the number of DCT blocks, when such number is large enough and these blocks are independent, based on the "Law of Large Numbers", the displacement of the ALV of a field will be small. The detailed demonstration is shown in Appendix A. Fig.1 illustrates one kind of method to divide the 640×480 format picture into 10 fields which is composed of $640\times480/8/8/10=480$ 8 by 8 blocks. The distance between any of the two blocks in the same field must be as far as possible to satisfy the independence requirement. Such a blocks partition philosophy is the same as cell allocation philosophy in cellular mobile communication system [3] and it is benefit for satisfying this independence requirement. By changing the ALV of the fields, we can embed 1 bit in each field. Since 480 are large enough to resist the "noise", the receiver can successfully detect the embedded bit with great probability.

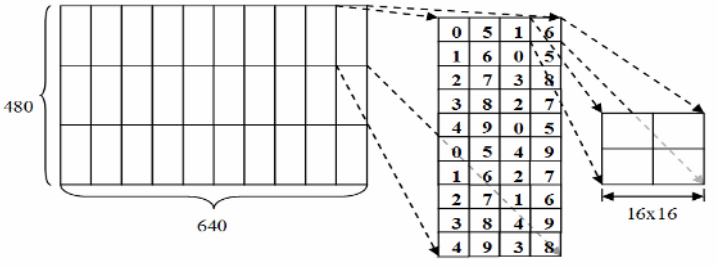


Fig. 1. One type of Block set partition mode in 640x480 format sequence

Although the adjustment can take place in both pixel domain and DCT domain, the DCT domain seems better because it is nearer to the output bitstream than the pixel domain and thus makes the embedding more precisely. In Appendix A, we have analysis the relationship between the ALV of a 8x8 DCT block (no matter residual block or intra block) and it's DC component after quantization then we have come to the conclusion: To increase the average value of a block by Δ in MPEG2 encoder, set

$$C_D^q = \left\lfloor (8\Delta + C_D + \frac{Q}{2})/Q \right\rfloor \tag{1}$$

Here the meanings of parameters in equation (1) are illustrated in Fig.2 which is the outline of DCT domain key embedding diagram using MPEG2 encoder. By using this DCT domain key embedding diagram, we can embed a key bit into each field of the picture.

The above processes have determined that the ALV of a field can be a carrier wave to bear the key data. Fig.3 illustrates the modulation scheme and detecting scheme we have proposed.

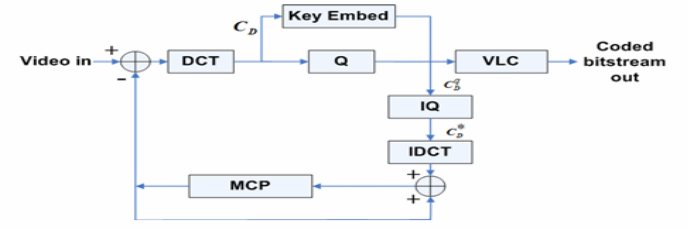


Fig. 2. MPEG2 encode diagram to change the average luminance value in the bitstream to embed data. MCP stands for motion-compensated prediction.

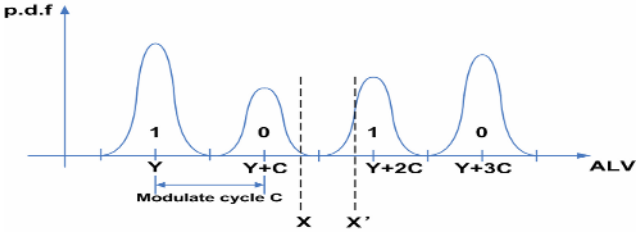


Fig. 3. Receiver's theoretical ALV distribution by using quantization index modulation

The horizontal coordinate is ALV (Average Luminance Value) of a field, Y is a real number arbitrarily selected between [0, 255] as an anchor value to prevent ALV value from being out of range and must be known by both the encoder and the receiver. C is the modulation cycle. Once Y is determined, the value of C depends on the distribution of ALV deviation after transcoding which is in inverse ration to the block set size N and in direct proportion ratio to transcoder's quantizers. In our experiments, C ranges from 4 to 16. Adding integer numbers of C to the datum mark Y will produce a series of modulation reference points Y+nC (n=0, 1…). Note that when n is an odd number, the reference point is for embedded bit '0', otherwise it is for embedded bit '1'. The reference points are the only values that the block sets' ALV can be after the embedding modulation. As shown in Fig.4, to embed a bit '1' in a region of a picture, we first compute the ALV value of this block set, assuming it is X, then we find '0' is the nearest reference point. So we change the ALV of this region to be Y+2C which stands for bit '1'. This ALV value might be slightly changed during the transmission. On the receiver side, decoder can compute the region's ALV value X', by judging the type of its nearest reference point, decoder can draw out the embedded bit easily. Here, the displacement of X' to Y+2C is due to the network handling and the transcoder. The probability of successfully detecting one bit key depends on modulation cycle C and the quantizers used in the transcoder.

2.3 Error Resilient Key Embedding Algorithm

The reliability of key data transmission can be improved by Forward Error Correction (FEC), thus lowering the key data missing probability in packet lossy environment. In this paper, we use Reed-Solomon (RS) codes [4]. For symbols composed of *m* bits, the

encoder for an (*n, k*) RS code group the incoming data stream into blocks of *k* information symbols (*km* bits) and appends n-k parity symbols to each block. And *n* is the block length. For RS codes operating on m-bits symbols, the maximum block length is : $n_{\max} = 2^m - 1$.For an (*n, k*) RS code, any error pattern resulting in less than $t_1 = \left\lfloor \frac{n-k}{2} \right\rfloor$ symbol errors can be corrected. For symbol erasures (symbol loss or known symbol errors), (*n, k*) RS code can correct $t_2 = n - k$ erasures. Generally, if

$$2t_1 + t_2 \le n - k$$

the RS code can recover all the original symbols. In this paper, we choose m = 8, therefore a symbol corresponds to one byte.

2.4 Key Data Frame Format

To make the system more robust, a key data frame format can be designed for synchronization of key data detecting. The key data stream will be sent in the form of frames, and the key technique used in this procedure is called “bit stuffing”. This technique makes the frame be able to contain any amount of bits. In this schema, each frame uses a binary sequence that consists of two ‘0’ and eight ‘1’, i.e., ‘01111110’ to denote the place of its beginning and end, and the key data is placed between the two flags. Since the key data stream may also contain the same sequence, the sender should count the number of “1” in series in the data stream. When the number reaches 5, a bit of “0” will be inserted, by what means the sequence “01111110” existed in the data stream becomes “011111010”. The receiver still counts the number of “1” in series. If a sequence of “01111110” is detected, the starting or end flag is met and the receiver determines to start or end storing a frame of data to its memory. Otherwise when “0111110” is detected, the receiver removes the last “0” automatically and saves the “011111” before it. In this scheme the border between two frames can be uniquely detected, since the flag sequence never exists in the data stream.

3 Simulation Experiments

As for source signals, we use two MPEG2 test sequences *dinosaur* and *Live-captured video* which are both encoded at 640x480 size and 20fps using 500frames. *Dinosaur* contains fast motion and scene change; while *Live-captured video* is derived from captured video, which contains slow motion and fixed scene. And our simulation platform is illustrated in Fig.4.

The source-coding distortion introduced by our key embedding algorithm is illustrated on Fig.5. Fig.5 (a) illustrates the PSNR of the dinosaur sequence at the receiver. We can find that when modulation cycle smaller than 4 can provide good performance and the distortion derived from key embedding can be neglected. Fig.5 (b) illustrates the probability of successfully detecting all the 10 bits in a frame changing with the modulation cycle C. Here we can see that when the successfully detecting probability is set to 0.98, the modulation cycle can be minimized with the maximum profit.

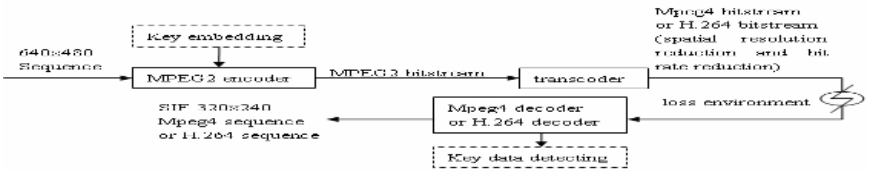
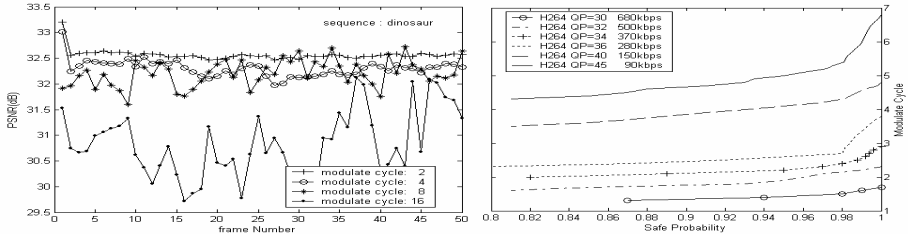


Fig. 4. Block diagram of key data embedding, transmission and key detecting



(a) (b)

Fig. 5. PSNR of frames and the probability of successfully detecting 10 bits in a frame changing with the modulation cycle C at the receiver

Assuming that the probability of successively detect a bit from a region is α, and the packet loss rate is β, then the probability that there is no key bit error in a frame is

$$p_{framesafe} = \alpha^{10}$$

The value of $p_{framesafe}$ is determined by the luminance modulated scheme. Practically, trading off between safety and video quality, we usually set $p_{framesafe} =$ 0.98. The probability that the parity bits can check out the key bit error in a frame is

$$p_{checked} = \sum_{i=0,1,3,5} \sum_{j=0,1,3,5} [P(i \text{ bits error in region } 1,2,3,4,9) \times P(j \text{ bits error in region } 5,6,7,8,10)] - p_{framesafe}$$

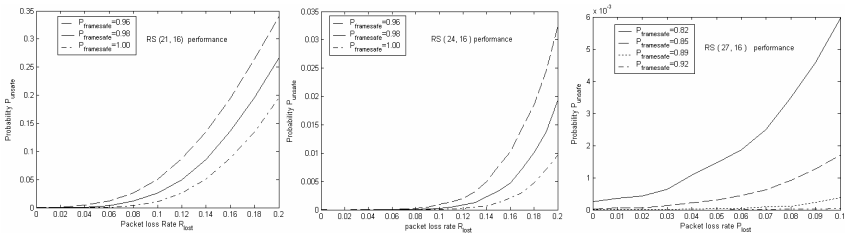
$$\approx C_{10}^{1}\alpha^{9}(1-\alpha) + C_{10}^{2}\alpha^{8}(1-\alpha)^{2}\frac{1}{2} + C_{10}^{3}\alpha^{7}(1-\alpha)^{3}\frac{1}{4} + C_{10}^{4}\alpha^{6}(1-\alpha)^{4}\frac{1}{2}$$

(ignore the advent of > 5 key bit errors in one frame), thus comes the probability of "erasure" for a frame $P_{erasure} = \beta + p_{checked}$ and the probability of "error" for a frame is

$$P_{error} = (1 - p_{checked} - p_{framesafe})$$

Fig.6 illustrates the probability of safely received 128 bits key data ***vs*** the loss packet rate of the network. It can be seen that the key data error rate of a GOP (P_{unsafe}) increases with increasing packet loss rate (R_{lost}) for different value $p_{framesafe}$.

This RS code can only be used when the network transmission is extremely reliable and the value of $P_{framesafe}$ approach to ideal ($P_{framesafe} \to 1$). Increasing the number of redundant packet by 3, we get the results indicated in Fig.6(b). In this situation, the target value $p_{framesafe}$ is quite satisfied when packet loss rate is lower than 16% and $P_{framesafe}$ = 0.98. As for the case when the value of $P_{framesafe}$ is especially high, we can again increase the number of redundant embedded frames, with results shown in Fig.6(c), to serve the high reliability of key drawing. The only flaw of such increasing is that the feasible rekey cycle becomes longer.



(a) (b) (c)

Fig. 6. The error rate of a GOP by using different packet loss rate and redundant packets

Table 1. Complexity of the proposed key embedding algorithm

Sequence	*Dinosaur*	*Live-captured*
Coding time without embedding	1.21f/s	1.28f/s
Coding time with embedding	1.12f/s	1.18f/s
Increased processing time	8.03%	8.47%

As for the embedding speed of the algorithm, table 1 presents comparison of coding times with/without key embedding. Note that the two sequences are all 500 frames with 640x480 pixel format and the CPU used is P4 2.0G.

4 Conclusion

In this paper, we propose a new key embedded video coding scheme, which is of error resilience and transparence for adaptation mechanism, and can be combined with existing key distribution mechanism to provide access control for video multicast applications. After a series of simulation experiments, we can see from the figures that when using more redundant embedded frames, the probability of successfully detecting the key data can be quite satisfied in a worse network environment.

References

[1] Wade Trappe, Jie Song, Radha Poovendran, and K.J.Ray Liu, Key Management and Distribution for Secure Multimedia Multicast, IEEE Transaction on Multimedia, Vol. 5, No.4, December 2003, pp.544-557.

[2] H. Yin, C. Lin, F. Qiu, B. Li, Z. Tan, A Media-Dependent Secure Multicast Protocol for Adaptive Video Applications, in Proc. of ACM SIGCOMM Asia Workshop 2005,Beijing China.

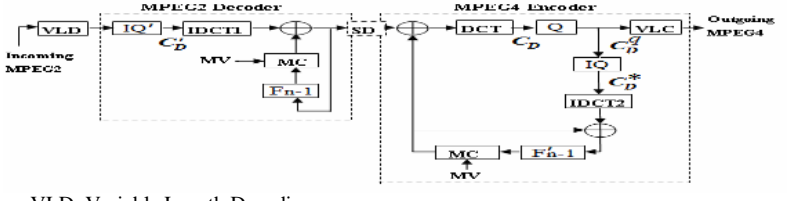
[3] D. Everitt and D. Manfield, Performance analysis of cellular mobile communication systems with dynamic channel assignment, *IEEE J. Select. Areas Commun.*, vol. 7, pp. 1172-1180, Oct. 1989.

[4] G. Solomon, Self-synchronizing Reed-Solomon codes (Corresp.), IEEE Transactions on Information Theory, Volume: 14, Issue: 4, Jul 1968, pp:608 - 609.

[5] P.Yin, M.Wu, and B.Liu, Video Transcoding by Reducing Spatial Resolution, Proceeding of the IEEE Int'1 Conf. on Image Processing (ICIP), Sept. 2000, Vancouver, Canada.

[6] N.Merhav, V.Bhaskaran, A Transform Domain Approach To Spatial Domain Image Scaling, HP Labs Technical report, HP-94-116, December 15, 1994.

Appendix A:



VLD: Variable Length Decoding
VLC: Variable Length Coding
MV: Motion Vectors
Q: quantize
IDCT: Inverse DCT
SD : spatial solution downscaling
MC : Motion Compensation
IQ: De-quantize
F : Reconstructed Frame

Fig. 7. Simply Cascaded Pixel-Domain Transcoder (MPEG2 to MPEG4)

The module SD (spatial resolution downscaling) in the transcoder diagram is optional, it converts the $M \times N$ spatial resolution into a stream with smaller resolution, such as $M/2 \times N/2$ or $M/4 \times N/4$. Usually, the downscaling can assure the average luminance value in a 16×16 luminance block remain unchanged after downscale to 8×8 luminance block no mater the downscale is operating in the spatial domain or in the DCT domain. Details can be found in [5, 6].

The input sequence to the MPEG4 encoder is a YUV sequence decoded by the MPEG2 decoder. Let C_D be the DC coefficient of an 8×8 block after discrete cosine transform in MPEG4 encoder, C_D^q be the coefficient after being quantized by Q_D, and C_D^* be the reconstructed DC coefficient of C_D, we have:

$$C_D^q = \left\lfloor (C_D + \frac{Q_D}{2}) / Q_D \right\rfloor,\ C_D^* = C_D^q \times Q_D$$

Let $D_Q = C_D^* - C_D = \left\lfloor (C_D + \frac{Q_D}{2}) / Q_D \right\rfloor \times Q_D - C_D$. Obviously, D_Q is uniform distributed between $-Q_D/2$ and $Q_D/2$, and the expected value of D_Q is

$$E(D_Q) = 0, \tag{2}$$

and the variance of D_Q is

$$Var(D_Q) = \frac{1}{12} Q_D^2 \tag{3}$$

Suppose that there are N times of such different blocks in a same frame, with D_Q marked as D_Q^i ($i = 1, 2 \ldots N$). Let $\overline{D} = \frac{\sum_{i=1}^{N} D_Q^i}{N}$, then we also have:

$$E(\overline{D}) = E(\frac{\sum_{i=1}^{N} D_Q^i}{N}) = \frac{1}{N} \sum_{i=1}^{N} E(D_Q^i) = 0 \tag{4}$$

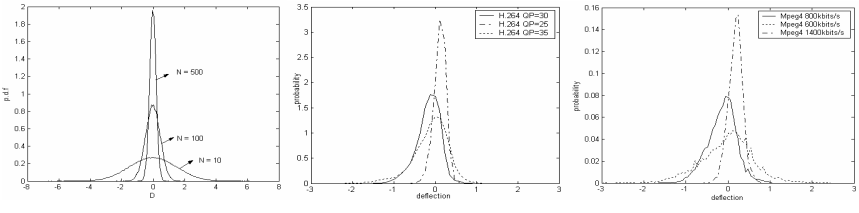
and

$$Var(\overline{D}) = E(\overline{D} - E(\overline{D}))^2 = E(\frac{\sum_{i=1}^{N} D_Q^i}{N})^2 = \frac{1}{N^2} E(\sum_{i=1}^{N} D_Q^i)^2 \tag{5}$$

if blocks are independent, then D_Q^i are independent, we have:

$$Var(\overline{D}) = \frac{1}{N^2} E(\sum_{i=1}^{N} D_Q^i)^2 = \frac{1}{N^2} \sum_{i=1}^{N} E(D_Q^i)^2 = \frac{Var(D_Q)}{N} = \frac{Q_D^2}{12N} \tag{6}$$

Equation (6) tells us that for a "region" composed of a certain large number of independent luminance blocks, the average luminance value of the pixels compound this region will be only little changed during the transcoding manipulation. The extent of this value alteration is in inverse proportion to the number of the blocks.



(a) (b) (c)

Fig. 8. ALV of the field during the transcoding manipulation with different number of blocks (a) shows the theoretical distribution of ALV displacement of a field composed of N macroblocks after transcoding with $Q_D = 16$. (b) and (c) illustrate the experimental distribution of ALV displacement after H.264 and MPEG4 transcoding respectively.

Appendix B: DCT Domain Luminance Alteration

A two-dimensional DCT is performed on small blocks (8 pixels *by* 8 lines) of each component of the picture to produce blocks of DCT coefficients. The coefficient corresponding to zero horizontal and vertical frequency is called the DC coefficient:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N} \quad (7)$$

Where:

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & for \quad u,v = 0 \\ 1 & otherwise \end{cases} \quad (8)$$

Substitute N by 8, for the value of DC coefficient of block is

$$F(0,0) = \frac{1}{8} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \quad (9)$$

From equation (9) we can come to the following conclusion:

Theorem 1: for 8 by 8 blocks, if each of the luminance pixel value is increased by ΔY, the DC coefficient will also increase by $8\Delta Y$. In other words, the increasing of DC coefficient by $8\Delta Y$ will lead to the average luminance value of an 8 by 8 block increasing by ΔY.

To change the average value of the block according to the volume of embedded data, operation can be performed on C_D^q which will be VLC coded in the MPEG2 bitstream. Here C_D^q is a DC coefficient value after alteration and quantization. Attention must be paid that the AC coefficients are not changed. As for the value of the reconstructed DC coefficient C_D^*, we can express it by:

$$C_D^* = Q \times C_D^q \quad (10)$$

According to theorem 1, the increase of the average luminance value of a block is:

$$\Delta = (C_D^* - C_D)/8 = (Q \times C_D^q - C_D)/8 \quad (11)$$

If we want to increase the average value of a block by Δ, we can set

$$C_D^q = \left\lfloor (8\Delta + C_D + \frac{Q}{2})/Q \right\rfloor \quad (12)$$

Although the average luminance pixel of a block could be changed easily after quantization, two troubles are still remained. First is that equation (12) does not promise to find out a C_D^q precisely change the average value we need unless $(8\Delta + C_D)/Q$ is an integer. Second is the limitation of the pixel value which prevents the exact change of the value even if $(8\Delta + C_D)/Q$ is an integer. For example,

increasing a block's average luminance value by 1 will be ignored if the luminance values of the block are totally 255.

Fortunately, the rigorous demand is not for a block, but for a region. Because of the large number of blocks, if we come up against unsatisfied block value change, we can compensate from other blocks. In this section, we forward a simple scheme to control the value change of the blocks in a region as shown in Fig.9.

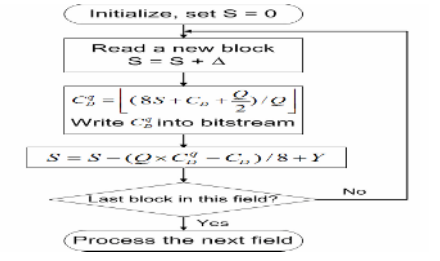


Fig. 9. Scheme to change the average luminance value of the blocks to embed data in the region. Δ is the average value increased in the region. Q is the quantizer of the DC coefficient. S is the residual ALV for all blocks that have been processed.