To appear in *IEEE Trans. Circuits & Systems for Video Technology*, Special Issue on Wireless Video, 2002.

A FORMAT-COMPLIANT CONFIGURABLE ENCRYPTION FRAMEWORK FOR ACCESS CONTROL OF VIDEO

Jiangtao Wen[1], Mike Severa[1], Wenjun Zeng[1], Max Luttrell[1], and Weiyin Jin[2]

***Abstract* – In this paper, we introduce new methods of performing selective encryption and spatial/frequency shuffling of compressed digital content that maintain syntax compliance after content has been secured. The tools described in this paper have been proposed to the MPEG-4 Intellectual Property Management and Protection (IPMP) standardization group and have been adopted into the MPEG-4 IPMP Final Proposed Draft Amendment (FPDAM) [9]. We will describe the application of the new methods to the protection of MPEG-4 video content in the wireless environment, and illustrate how they are used to leverage established encryption algorithms for the protection of only the information fields in the bitstream that are critical to the reconstructed video quality, while maintaining compliance to the syntax of MPEG-4 video, and thereby reduces the amount of data to be encrypted and guarantees the inheritance of many of the nice properties of the un-protected bitstreams that have been carefully studied and built, such as error resiliency, network friendliness, etc. The encrypted content bitstream works with many existing random access, network bandwidth adaptation and error control techniques that have been developed for standard-compliant compressed video, thus making it especially suitable for wireless multimedia applications. Standard compliance also allows subsequent signal processing techniques to be applied to the encrypted bitstream.**

Index terms **— wireless multimedia distribution, digital rights management, and encryption**

I. INTRODUCTION

Access control is a central piece of an intellectual property protection system. One of the major goals of content access control for entertainment purposes (as opposed to for top secret communications, e.g. protection of military information, where end to end encryption is usually required and a secure perimeter is usually established to improve overall system security) is to enable authorized users to view the video, and to disallow unauthorized users to view the video with satisfactory quality, similar to a set top cable TV scrambler. One common method for access control is through encryption.

Encryption of multimedia content in an access control system is not always simply the application of established encryption algorithms, such as DES [11] or AES [12], to content bitstreams. A number of issues need to be considered when selecting the proper encryption technique for an access control system. The first consideration is identifying a protection level that is appropriate for the content. It is widely understood that not enough protection will result in too many pirated copies and lost sales, and will degrade the value of the legal copies, but it is also important to realize that over protecting the content will not only increase the cost, but make the product too difficult to use, and may offend the user. It may also result in too *few* legal copies, which might also do harm to the business model. As pointed out in [6], for many real-world applications such as pay-per-view, although the content data rate is very high, the monetary value to the bits is low; therefore "very expensive attacks are not interesting to adversaries" and "hence, light-weight encryption algorithms which can provide sufficient security level and have an acceptable computation cost are attractive to MPEG video applications".

Other factors that also need to be factored in include platform, bandwidth, distribution channel, application, and if any signal processing will need to be carried out before the content reaches the end user. In addition, the need for the encrypted content to be decrypted, processed, and then re-encrypted has to be minimized. This is because decryption/re-encryption introduces significant processing overhead, and necessitates encryption/decryption capabilities in various modules. The latter is highly undesirable given the wide spectrum of encryption algorithms in use today, many of which are proprietary. From a content provider's point of view, inline decryption/re-encryption surfaces clear content before it reaches the end-user, and poses significant security threats.

A number of international standards bodies working on standardizing intellectual property protection technology or

[1] PacketVideo Corp., 4820 Eastgate Mall, San Diego, CA 92121.
[2] EE Dept. Univ. of Southern California, Los Angeles, CA

frameworks in a number of areas, such as the Motion Picture Expert Group (MPEG) [9], and Electronic Book Exchange (EBX) [10] have realized that a one-size-fits-all solution to access control will not meet the requirements for their target applications, especially, it is not always desirable, or even feasible, to perform a "wholesale" encryption of the entire content bitstream, as would be the case in SRTP [13]. MPEG-4 Intellectual Property Management and Protection (IPMP) [9] took the approach of providing an interoperable framework with normative messages that can be used to select and configure the most effective and appropriate tools for the protection of content that is most appropriate for the specific application. The framework is also capable of accommodating various existing and forthcoming security algorithms and protocols, and security requirements.

In the next section, we will introduce two new methods of securing multimedia content that have been adopted by MPEG-4 IPMP [9]. The tools encrypt or shuffle variable length code (VLC) codeword concatenations and at the same time maintain full bit level compliance to content syntax (in this case MPEG-4 video). We will discuss the necessities for these tools in the wireless environment and details of their applications to wireless multimedia and then present some simulation results and discussions, and some conclusions.

II. A FRAMEWORK FOR CONFIGURABLE FORMAT-COMPLIANT CONTENT PROTECTION WITH SELECTIVE ENCRYPTION AND SHUFFLING

Encryption of video content in the compressed domain can be achieved in various ways, the simplest of which is to encrypt the entire compressed video bitstream with a cipher, similar to the approach taken in Secure Real Time Transport Protocol (SRTP), in which the entire payload and some packet header information is encrypted and then the encrypted information and the unencrypted part of the packet header information necessary for correct handling of the packet are authenticated [13].

For various reasons mentioned in the previous section, simplistic direct encryption of content bitstream may not be the most desirable in all applications for all contents. For many applications, only selective encryption of content bitstream is required or desired.

The core idea of selective encryption is to encrypt and/or shuffle only a portion of the compressed bitstream so that the resultant bitstream is not decodable or the resultant quality becomes unacceptable for the target application without "unscrambling" of these fields. The security of the system can be judged by the working factor needed to "crack" the system and the associated cost, as compared with the content value and expected shelf-life.

Earlier schemes of selective encryption of compressed video bitstream involved encryption of only header (e.g. MPEG-2 Group of Picture headers, MPEG or H.26x slice headers and macro block (MB) headers, etc.) and provided only nominal security, as headers do not contain much information. Encrypting only I frames is also not very secure (i.e. work factor for "cracking" the system is low), as a result of the Intra refresh in standards such as MPEG-4 or H.26x [14].

Later selective encryption schemes combine encryption of headers with I frames or other critical information (e.g. DCs or low frequency ACs, etc.), and might introduce delay or overhead [3]. In [5], a random frequency domain shuffling of DCT coefficients is performed before entropy coding, which will result in a compliant "encrypted" bitstream, however, as the shuffling destroys the inherent energy packing capability of the transform, compression efficiency is compromised. This is especially true for low bitrate video, where the few non-zero coefficients tend to cluster locally, but might be shuffled apart as a result of encryption.

In the Real-time Video Encryption Algorithm (RVEA) of [6], at most 64 motion vector and/or DCT sign bits are encrypted for each encrypted macro block (MB). These include *at most* 64 sign bits for motion vectors (for P and B frames) and the non-zero DC and the lowest frequency non-zero AC coefficients (for I, P and B frames). Though this resulted in an *upper bound* to the amount of data to be encrypted, and effectively reduces the encryption complexity, it had the potential problem of not having a *lower* bound of the amount of data encrypted, which is related to the level of security provided by the scheme. This is especially true for low bit rate or wireless video, where high quantization results in a large amount of skipped MBs or 8x8 blocks where no sign bits could be located.

The above schemes were designed based on knowledge of the relative importance of different fields. In some systems, to further reduce complexity, a syntax-unaware "run-length" based selective encryption (SURLSE) is carried out, e.g. an encryption of X consecutive bits is followed by Y bits that are not encrypted, which are followed by another Z encrypted bits, and so forth. One potential security problem of SURLSE is that the encrypted bits may not be the most critical bits in the compressed video bitstream. From a security point of view, encrypting low entropy fields in the bitstream such as markers, start codes, time codes, etc. does not provide security. Given the same amount of data to be encrypted, it is highly desirable to encrypt the fields in the content bitstream that are most critical to content reconstruction. The problem is more serious when the amount of bits that need to be encrypted is small, such as the case for any wireless multimedia systems. On the other hand, wholesale direct encryption (which can be considered as a special case of SURLE)

creates marker and header emulation. Markers and headers are special bit patterns in a compressed bitstream used for synchronization purposes and are usually designed so that they cannot be emulated by valid concatenation of other bits in a compressed bitstream. Marker and header emulation can introduce problems when the encrypted content is transmitted over certain network protocols designed for unencrypted compressed bitstreams. For example, the Motion Marker in the data partition mode of MPEG-4 video syntax [4] is 17 bits and is not byte-aligned, therefore, on average it can be emulated in every 2^{17}=131072 cipher text bits, which is not a large number of bits for even low bitrate video. For example, 131072 bits is about 8 seconds' worth of video at 16kbps.

The configurable selective encryption framework of MPEG-4 IPMP FPDAM [9] supports SURLE, encrypting only headers, or any particular type of frames (e.g. all I frames), or any particular fields in any particular type of frames. Most of such encryption will result in a non-compliant bitstream after encryption has been carried out.

For many applications, however, maintaining bitstream compliance after encryption is very important. Due to several unique characteristics of the wireless networks and the relatively low processing power and memory of mobile terminals, content encryption for wireless multimedia applications has a number of unique challenges and is one of the applications to which syntax compliance after content encryption is of importance in many circumstances.

Complexity and security trade off: For wireless multimedia content encryption, especially for consumption on mobile terminals, low processing overhead becomes an extremely critical requirement. Due to the small amount of data that could be encrypted/decrypted given the tight processing power and memory budget, it becomes important to "select" the most critical bits to encrypt. Syntax compliance makes it easier to locate encrypted fields in protected content without side information, and is therefore desirable.

Network adaptation and friendliness: Given the time-varying nature of some wireless transmission channels, to guarantee quality of service (QoS) for wireless multimedia applications, on-line dynamic bandwidth adaptation usually needs to be performed under stringent delay and cell loss constraints on already packetized, and therefore encrypted content bitstreams. Therefore it is desirable to make the encryption transparent to network adaptation processes such as transcoding or rate shaping.

Error resilience: Handling of transmission errors and losses is also very critical for wireless multimedia, and great efforts have been dedicated to the design of digital content compression standards to provide some level of error resiliency. It is desirable that most of these existing tools can be readily applied to the transmission of an encrypted video bitstream.

Aside from the above requirements, issues such as multiple levels of security, proof of security, and applicability of various signal processing such as compressed domain watermarking, random access, scene change detection, content-based searching or filtering, etc. to the encrypted bitstream (without decryption) in the various links of an end-to-end chain [1][2] are also of great concern to wireless multimedia.

The ability to maintain format compliance provides *ONE* satisfactory solution to all of the requirements discussed above and in Section I, even though many solutions do exist that handle one or several issues raised. By reducing the amount of data to be encrypted/shuffled to only "important" information carrying fields of a compressed bitstream, it provides a light-weight solution for access control. Syntax compliance inherits network friendliness, error resilience as well as the possibility of performing various types of signal processing on the encrypted bitstream directly. In addition, the encrypted bitstream will work nicely with protocols designed for the transport of standards-based compressed bitstreams. There will be no marker emulation problem. A standard player that is not aware of the encryption will not crash, so an old system where no DRM solution was included can deal with secured content gracefully.

In the following sub-sections, we will describe in details format compliant selective encryption [2] and compliance preserving secure spatial shuffling [15].

A. Format Compliant Selective Encryption

Selective encryption of compressed video bitstreams by itself is not a new idea. Techniques proposed in some of the previous work [1], [5], [6] could result in an encrypted bitstream that still conforms to the standard. However, the importance and value of maintaining standard compliance has not been generally recognized except for in [3], [7], [8], where syntactical logic structure is preserved in a way that is outside the scope of syntax, and in [1] where the features such as processing overhead, data selectivity, error resiliency, different levels of security, transcodability and applicability of signal processing without decryption were discussed to some extent in a joint encryption and compression framework. Because direct encryption of VLC codeword concatenations will usually not result in a valid VLC codeword concatenation, and different fields in a compressed bitstream often exhibit dependencies that need to be validated, the encryption that can be carried out in these previous schemes are usually very limited in scope and were restricted to mostly simple fixed length fields, such as DCT or MV sign bits.

The main contribution of this paper is the introduction of a framework and new tools that could achieve any level of syntax compliance in any format/standard through a unique compliance-preserving encryption method of variable length coded fields in compressed bitstreams.

A compressed video bitstream can be divided into information-carrying and not information- carrying portions. For video content compressed using standards such as MPEG or H.26x, fields such as markers usually have a fixed pattern, and their locations can often be determined based on other information carrying fields in the bitstream. Therefore, from a security point of view, these fields do not really need to be encrypted. The information carrying fields are what need to be encrypted. For an MPEG or H.263 compressed video bitstream, these fields are selected fixed length code (FLC) or variable length code (VLC) codewords with different levels of importance to perceived quality.

To achieve format compliance, we extract bits from the fields that we've chosen to encrypt, concatenate them in an appropriate way, encrypt the concatenation using a public key encryption algorithm or a symmetric algorithm such as DES, AES, or SEAL [16], and then put the encrypted bits back into their original positions. Because many of the information carrying fields are coded with VLCs and for any given length, not all possible combinations of bits represent valid VLC codeword concatenations, simplistic encryption of bits may not result in another valid codeword concatenation, and in which case, will not result in format compliance. To illustrate the problem, suppose we have a 4 codeword table 0, 10, 110, 111, and a two-codeword concatenation of 010. If we encrypt the codeword concatenation directly, it may result in a bitstream of 001, which does not correspond to any valid codeword concatenation.

To maintain compliance, we've developed a way to encrypt a concatenation of codewords from a VLC code table, such that it is secure, and that the bitstream after encryption still contains a valid concatenation of codewords with exactly the same number of codewords from the same code table. This technique, when applied appropriately to compressed multimedia content in conjunction with other tools described in the paper, achieves security while maintaining compliance to the syntax.

To better explain the idea, we use MPEG-4 video [4] as an example. Figure 1 illustrates the process of encrypting the motion vector information for one MB. In MPEG-4, each frame is divided into MBs of size 16x16. Each MB consists of four 8x8 luminance blocks, and two 8x8 chrominance blocks. Each MB bitstream consists of header information that signals whether the MB is coded, and if it is coded, its coding mode (INTER or INTRA and so forth), the number of motion vector pairs (either 1 or 4) if the MB is INTER coded. Finally the header also contains information about which of the 8x8 blocks within the MB has DCT information. Bits for consecutive MBs are concatenated and form a video packet, which is delimited by a marker or start code. The bitstream for the DCT information of an 8x8 block consists of VLC codewords for (RUN, LEVEL, LAST) information (called EVENTS), with "LAST" indicating the end of an 8x8 block bitstream.

The technique works as follows for a VLC table with N codewords, where N is the *n-th* power of two (i.e. $2^n = N$). Before encryption, a fixed length *n-bit* index is first assigned to each codeword in the VLC code table. Then after a concatenation C of codewords from the code table is obtained, a bit string S is constructed by concatenating the indices for codewords contained in C. Because different types of fields are often interleaved, obtaining concatenations of codewords from the same table may involve parsing the bitstream and constructing concatenations of codewords not contiguously present in the bitstream. S is next encrypted with a chosen secure cipher operating in a chosen mode deemed suitable for the content, application, network and device (Fig. 1). The string of bits after encrypting S, denoted S', is then mapped back to codewords in the code table (which can form a concatenation C' of codewords) using the same index-to-code-book-entry map. Codewords from the C' are then put back into the content bitstream in place of the original codewords in C.

In decrypting VLC codewords encrypted using the above technique, the exact opposite operation is carried out, i.e. the encrypted codeword concatenation C' is obtained by parsing the bit stream and extracting the codewords. These are then mapped to an encrypted index sequence, S', which is decrypted to index sequence S, and then mapped to codeword concatenation C, and from this concatenation the original codewords are put back into the content bitstream.

Note that to guarantee that C' has exactly the same number of codewords as C, the cipher should be chosen so that the length of its output (in bits) is identical to the length of its input. This property can be easily maintained by using block ciphers such as AES and DES in the appropriate mode (e.g. cipher text stealing mode) or by using stream ciphers. Padding with "dummy" data for block ciphers should usually be avoided, unless warranted by the particular application, for example, in which the number of encrypted codewords does not have to be identical to the number of codewords before encryption.

When N, the total number of codewords in the VLC table T, is not a power of 2, the table can be divided into non-overlapping subsets of T, T_1, T_2, ..., T_m, with N_1, N_2, .., N_m codewords respectively (different N_i's do not have to take on different values), each being a power of 2. Then when

code word concatenation *C* is obtained, it is mapped to an index concatenation *S* by concatenating indices of codewords into the corresponding subset T_i to which the codeword belongs. For example, if in *C*, a codeword *X* from T_i with 8 codewords is followed by a codeword *Y* from T_j with 4 codewords, then the corresponding index concatenation in *S* will be the 3-bit index for codeword *X* in T_i, followed by the 2-bit index for *Y* in T_j. Then the same encryption can be carried out on S, and the encrypted index sequence S' can be divided in a similar way and mapped to codewords.

It should be noted, however, when this approach is taken, the design of the sub-tables should be carefully carried out so that the size of each subset is sufficient for security. The design of the sub-sets also impacts the difference in length (in bits) between *C'* and *C*. As a general guideline from the security perspective, the largest subset of the original table should consists of the most likely subset of codewords, so that the effect of subset indexing is least "visible" to an attacker.

Because of the randomizing effect of ciphers, the length (in bits) of *C'* will be different from the length of *C*, with the length of *C'* on average longer, even though both *C'* and *C* contain the same integer number of codewords from the same code table. More discussions on this issue can be found in the following section.

Because fixed length codes are just a special case of variable length codes, the exact same approach above can be carried out when encrypting FLC fields in content bitstreams. If the code table has a total number of codewords that is a power of 2, then each codeword itself can be regarded as the index to the codeword, and the codeword concatenation *C* and the index concatenation *S* become identical. In this case, the "map to index" and "map back to codeword" steps can be skipped. However, when 1) the total number of codewords is not a power of 2; or 2) if one only intends to encrypt a subset (with a power of 2 number of codewords); or 3) if one desires to use indices for FLC codewords that are different from the codewords themselves, the mapping to index and back steps must be performed.

When forming the concatenation and indexing codewords, one might also interleave codewords from different "logical units" of the original media content bitstream when constructing *C*, and/or interleave indices for different fields using different tables when constructing *S*. One possible example of this extension is for MPEG-4 video, one may want to encrypt INTRA MB DC information, together with INTER and INTRA block DCT sign information and INTER MB motion vector (MV) information. To do this, one may use a 5-bit index for DC, the 1-bit DCT sign as index to itself, and a 6-bit index for MV to index the codewords for these fields separately. The indices can be interleaved in the order in which the un-encrypted codewords show up in the bit stream. After encryption, the index sequence will be "broken" up into indices for different fields (e.g. in the previous example, 5-bit index for DC, followed by 1-bit indices for DCT signs, followed by 6-bit indices for MV), and then mapped into codewords and put back into the content bitstream. As an alternative to indexing codewords from different field separately, one can also produce a "master" code table by exhausting all valid combinations of codewords from tables for individual fields, from which indices can be determined for all combinations of the selected fields.

The above technique can be used with any media type (video, audio, image, graphics, text, data) to achieve the optimal tradeoff between application requirements and security. In designing the proper system for a given media type, syntax, application, platform, media value, and other requirements, one should carefully choose the fields to be encrypted, how they are concatenated, and proper cipher.

B. Spatial Shuffling of Codewords in Compressed Bitstreams

1. Basic idea

The basic idea of this approach is to spatially shuffle codewords of the compressed bitstream in a way such that the resultant bitstream complies with the compression format as much as possible, and is secure from attacks. Since it is simply a reorganization of codewords within a compressed bitstream, with the structure information (header/marker, etc.) intact, no bit overhead is incurred, as opposed to the approach of shuffling transform coefficients proposed in [1].

To better illustrate the idea, we again use MPEG-4 video as an example. In the following discussions, we will refer to a collection of codewords that are to be shuffled together as a *basic shuffling unit*. For example, a basic shuffling unit could be a single VLC or FLC codeword, or a code-stream corresponding to an 8x8 block or an entire MB. We will divide the bitstream into groups of basic shuffling units, where each group may have a different number of basic shuffling units and the basic shuffling units of different groups are to be shuffled separately using a shuffling table. The spatial boundary for each group (referred to as *clear-text unit* in this paper) could be a video packet boundary, a few rows of MBs, or a frame. The followings are some special cases that warrant some discussion.

MB as basic shuffling unit: in this case, the bitstream for the information corresponding to one MB will be shuffled as a basic unit. Since an MB bitstream is a self-contained unit, the resultant encrypted bitstream will be fully compliant to the MPEG-4 format.

Coded 8x8 block as basic shuffling unit: in this case, a coded 8x8 block bitstream will be used as a basic shuffling unit. No MB level information such as MV and MCBPC/CBPY (variable length codes that are used to derive the macroblock type and the coded block pattern) will be involved. Again, since each coded 8x8 block bitstream is a self-contained unit, the resultant scrambled bitstream will be fully format compliant. However, because INTRA and INTER DCT information are coded with different VLC tables, INTRA block bitstreams and INTER-block bitstreams should be shuffled separately, if full format compliance is desired. If all 8x8 blocks, either coded or not coded, of coded MBs are subject to shuffling, then the MCBPC/CBPY needs to be modified accordingly to make sure the resultant bitstream is format compliant. Note that this modification will have negligible impact on the bit rate. Note also that since intra DCs are often coded with codes of known length, they can be encrypted using common encryption algorithms as described in Section II.A, instead of being involved in the coded block shuffling process, i.e., the intra-DC can stay in its original block position while all other run-level codewords of a block will be shuffled.

Run-level codeword as basic shuffling unit: the run-level codewords within a group will be shuffled amongst each other. Let us denote the coded bitstream for an 8x8 block as {DC (for intra block only), RL_0, RL_1, … , RL_i, …RL_{last}}, where RL_i is the run-level codeword associated with the i-th non-zero coefficient in the block. We can group run-level codewords from different 8x8 blocks together according to their codeword index. For example, any run-level codewords whose codeword index is in the range of [0, T_1) will be grouped together and shuffled using one shuffling table. Similarly, any run-level codewords whose codeword index is in the range of [T_1, T_2) will be grouped together and shuffled using another shuffling table, and so on. Two special cases are grouping based on the index range of [0, 63], and a series of grouping based on the index ranges [0,0], [1,1], [2,2], …, [T, T]. The former case will result in the largest shuffling space, while the latter case results in the smallest shuffling space for each individual group.

It should be pointed out that different 8x8 blocks usually have different numbers of run-level codewords. The number of run-level codewords for each block should remain the same before and after shuffling in order to allow de-shuffling to work. In other words, the boundaries between blocks need to be preserved. This is achieved by making sure that the "last" field is preserved. When shuffling a coefficient that is in the last position in a block, and thus has its "last" field set to '1', it is possible that that coefficient is shuffled into the same block or some other block where it is no longer in the last position. In that case the "last" field must be set to '0'. Conversely, it is possible that some other coefficient that was originally not in the last position gets shuffled to the same or some other block and ends up in the last position in that block. In that case the "last" field must be set to '1'. Since the "last" field is not separable from the rest of the codeword, changing its value will require changing the entire codeword, which on average will introduce a slight, usually negligible, overhead. Another way to deal with this issue is to never shuffle the last run-level codeword in a block.

The sign bits of the run-level codewords are not necessarily to be shuffled, in other words, all the sign bits can stay in their original positions, and can be subject to encryption instead of shuffling.

Since different frames/MBs have different levels of importance to visual quality, in order to reduce the complexity, the spatial shuffling can be applied only to selected portions of the bitstream such as I frames, Intra-coded blocks, and a selected subset of the run-level codeword index range (e.g., only the first T run-level codewords of each block).

For run-level codeword based shuffling, a few interesting observations can be made: 1) the scrambled bitstream resembles an MPEG bitstream. As a result, most operations that work on MPEG bitstreams, e.g., transport packetization, frame dropping etc, can still work nicely on the scrambled bitstream. However, without de-scrambling, the number or position of decoded coefficients for some 8x8 blocks may exceed 64. An "error-resilient" decoder should be able to handle this situation, though, since it knows where an 8x8 block bitstream ends based on the "last" flag. Note that it is possible to generate a fully format-compliant scrambled bitstream by truncating some of the last un-shuffled run-level codewords, albeit at the cost of some video quality degradation; 2) Some flexibility for compressed-domain signal processing without deshuffling may be lost, due to the loss/uncertainty of the actual frequency/spatial location of each decoded coefficient. For example, requantization and watermarking are still doable in the transformed domain, but may be adversely affected to certain degree (because quantization step size could be different for different frequency bands and for different MBs); and 3) synchronization in the case of channel error. The shuffling table size for each group of codewords is group-dependent. Since the number of run-level codewords for each 8x8 block bitstream remains unchanged after scrambling, when there is NO bit error in the scrambled bitstream, this information will be available for calculating the shuffling table size needed for de-scrambling. However, If there are some channel errors, the number of run-level codewords for some 8x8 blocks may be erroneous, thus de-scrambling could be impossible. For packet-based network, one way to reduce this adverse effect is to shuffle only within a video packet (or a slice) so that if a packet is lost, error concealment will be applied, as opposed to de-scrambling.

2. Encryption-based self-synchronous shuffling table generation

Content access control by shuffling is usually subject to plaintext attack, where the attacker can reverse engineer the shuffling table if he has access to both the plaintext and cipher text. As a result, it is important to be able to change the shuffling table over time, e.g., from one shuffling group to another.

One way to achieve this is to constantly change the key used in generating the shuffling table. This may pose significant burden on the key management system, especially in the presence of packet loss. A better approach, as is described in the following, is to generate the shuffling table based on encryption on some "local" information that is not involved in the shuffling. Shuffling based on tables generated this way is self-synchronous, which is highly desirable in the presence of packet loss.

To this end, shuffling tables are generated or updated based on the cipher text output of applying a standard cipher (e.g., DES) to some bits specific to the local (spatially as well as temporally) information. These local-content-specific bits could be any bits in the local bitstream that are NOT going to be involved in the shuffling process (therefore are available prior to de-shuffling). For example, if we want to secure INTRA and INTER MB DCT RUN/LEVEL information through shuffling, then, a good candidate for the "local bits" that can be used in generating the shuffling table is the DCT signs, as they are easily available, usually very randomly distributed, and not to be involved in shuffling (could be encrypted instead). When MB or block is to be used as a basic shuffling unit, other local information bits such as the access unit sequence number, time stamps, if available, etc can be used.

Since the local-content-specific bits are local and varying, the resultant shuffling tables will be updated as well. The shuffling tables are resistant to plain-text attack, as long as the standard cipher used in the table generation process is resistant to plain-text attack.

The following is a very simple approach that can be used as an example to illustrate the idea of encryption based self-synchronous shuffling table generation. Here again, we use MPEG-4 video as an example, and assume that certain fixed length coded fields, such as DCT signs and INTRA DC information are encrypted using DES (in ECB mode which is self-synchronous) with key K_F, but not shuffled, while Run-Level codewords are shuffled based on a shuffling table generated using the following process.

a. We use some ciphers such as SEAL [16] (based on a less frequently changed key K_L, e.g., a fixed key for the whole video) to generate a random bit sequence R_L of sufficient length L (L should be larger than any *bit_length**K where K is the number of codewords in a shuffling group, and *bit_length* is the smallest integer that is no less than $\log_2(K)$). The same R_L will be repeatedly used for all shuffling groups within the lifetime of the key K_L. As will be seen, R_L will serve as a "master" random sequence to make sure the space of the shuffling tables generated in Step *d* is sufficiently large.
b. Then for each shuffling group with K RUN-LEVEL codewords to be shuffled, collect the K key-K_F-encrypted sign bits of the codewords. Denote the concatenation of these K sign bits as *R'*, we will encrypt *R'* again using DES (in ECB mode) with a dedicated key K_T, and denote the output as *R*. We then repeat *R bit_length* times to get *Rr* (i.e., Rr=RRRRR...R). The repetition also limits the computation to only encrypting K bits. Note that *R* is only used to generate the shuffling table. It is *never* in the clear without knowing the key K_T, as opposed to *R'*.
c. XOR R_L and *Rr*, the output is denoted as *Rc*
d. Divide *Rc* into K non-overlapped segments, each with *bit_length* bits. Create a shuffling table where each input index value *i* (from 0 to K-1) is mapped to an output index value determined by the *i*th segment of *Rc*
    i. In the case that K is not a power of 2, the output index value will be converted to that index value modulo K so that it lies in the range [0, K-1].
    ii. In case two or more input index values are mapped to the same output index value, we will re-assign, in a pre-determined order, un-used output index values (in the range from 0 to K-1) to those conflicting input index values.

Note that, given a fixed R_L, the possible number of shuffling tables is only 2^K (determined by *R* of length K). However, the attacker does not know which subset (of size 2^K) of the superset of $2^{Bit\_length*K}$ tables is used, because R_L, with a length of *bit_length**K bits, is unknown. As a result, for exhaustive search attack, the number of trials the attacker needs to conduct is the lesser of $2^{Bit\_length*K}$ (approximately equals K^k) and K!, which is always K!. The scheme is also resistant to plaintext attack, since *R* is generated based on DES encryption which is resistant to plaintext attack, and *R* is never in the clear without knowing the key K_T and K_L (even though the attacker may know some plaintext, i.e., both R' and *Rc*). The complexity involved in this shuffling table generation process is encryption of K bits, which is much less than straightforward encryption of K run-level codewords.

III. SIMULATIONS

To study the effectiveness of the proposed methods used under the MPEG-4 IPMP framework in the protection of wireless multimedia content, we used MPEG-4 video in error resilient mode with data partitioning [4] in our simulations. In this mode, the VLC-coded MB coding type information and motion vector (MV) information (header partition) is separated from the texture information for each video packet by the aforementioned 17-bit motion marker. The texture information is also VLC-coded, with a 1-bit "sign" field present in all codewords. Video packets are delimited by a byte-aligned 17-bit resynchronization marker, and contain a fixed-length index-to-first-MB information field to provide additional error recovery and error detection capability.

Format compliant selective encryption

In our simulations, we chose the FLC coded DCT sign, DQUANT, and INTRA DC information as the candidates for FLC encryption, and motion vectors (MVDs) for VLC encryption. For encryption of MVDs, each motion vector codeword is first assigned a fixed length index. The indices for each codeword are concatenated to form a sequence of indices S, which is encrypted with a cipher to result in encrypted motion vector index sequence S'. S' is mapped into a new set of MV codewords and these codewords replace the original codewords. At the decoder, S' can be constructed and then decrypted using the proper key, and the original motion vector index series S is recovered.

In our simulations, we tested 3 different configurations of increasing complexity and security: the lowest complexity/security configuration is to encrypt only the FLC fields identified above (therefore with no overhead in bitrate); the mid complexity/security configuration is to encrypt only the VLC coded MV fields (with a bit overhead dependent on the content and bitrate), and the highest complexity/secure configuration is to encrypt both the FLC and VLC coded fields (with the same overhead as in the second configuration). The encryption cipher was DES. FLC and VLC were encrypted separately when both were encrypted, as they were partitioned in the original syntax and may be subject to different processing/dropping during transmission. Each video packet was also encrypted separately, because video packets can be mapped to transport packets and thus might be dropped as units in a real network. The key management system used is out of the scope of this simulation. We only note that the same MPEG-4 IPMP FPDAM provides normative ways of sending authentication and key management information and data between IPMP tools and between tools and the terminal [9]. However, the scheduling of the keys and the synchronization of keys and packets will be implementation dependent.

Our simulations were focused on the trade offs between complexity, security and overhead with different configurations. We used 3 different sequences in the simulations and for each sequence, a number of different bitrates and frame rates. Among the 3 test sequences, "*Soap City*" is a sequence with medium to high motion and many scene changes, "*Hanging Up*" is a movie trailer with fewer scene changes and high motion, and "*Head*" is a talking head sequence.

A full fledged cryptographic analysis, though possible, of the security of the various configurations would not fit into the scope of this paper, however, we observe that the fields that were selected are critical to the correct interpretation of compressed video data. Fields such as DCT signs are also very random and are therefore secure from brute force guessing. Therefore, if the proper data encryption algorithm is used following general security guidelines, it could be assumed that *direct* cryptographic attack of the encrypted fields is impractical (i.e. not worthwhile for the retrieval of entertainment content).

We found that for sequences such as movie trailers, the decoded video usually looked scrambled enough to prohibit use for entertainment purposes after encrypting only the FLC fields. However, because motion information was not encrypted, the correspondence between MBs in consecutive frames was preserved, and certain content related information could sometimes be identified (e.g. when a scene change happened, when there was global motion, etc.). In contrast, when only VLC coded MVs were encrypted, the resulted content looked scrambled enough for most of the time, however, MBs that were coded as INTRA (and thus don't have any motion information), were not scrambled, which results in those regions of the frame being momentarily clear. However, because motion information was scrambled in subsequent frames, the clear regions disappeared quickly. Encrypting both FLC and VLC fields seemed to be immune to the problem of encrypting either of the two alone, and we believe it could offer a higher level of security and scrambling of content at the cost of more computation. Figure 2 shows screen shots of "Soap City", encoded at 160x112 resolution at 5fps/40kbps, with and without encryption. The overhead for encrypting the VLCs in this sequence is just under 9%.

The overhead resulting from applying the standard compliance encryption method to MVD information is included in Table 1, where "MaxMV *N*" indicates that only MVDs within the range of [-*N*, *N*] (measured in half-pel) are encrypted. The overhead is in general lower for bitstreams with higher bitrate, lower framerate, more scene changes, and more INTRA frames, where motion information does not take a large portion of the overall bitrate. For low bitrate, low motion (e.g. talking head) or high motion but with high frame rate sequences (e.g. *Soap*

City or *Hanging Up* at 10fps), because motion information does take a significant portion of the bitrate, the overhead will be higher. It should be recognized that although the introduced overhead is not unique to format compliant encryption introduced in this paper (e.g. schemes in [3]and [5] also introduce overhead), it is a drawback of the proposed framework, and therefore should be a primary concern when designing a selective encryption algorithm using this framework. The decision should be made based on the level of security that needs to be achieved, the type of the content, and target network's bandwidth.

Spatial shuffling of codewords

We tested the spatial shuffling idea using each video packet (of all frames) as the *clear-text unit* for shuffling. This configuration provides good error resiliency, but with the compromise of lower level of security. When some video packets only contain a small number of MBs, the shuffling space may not be sufficiently large. As a result, the scrambling effect and the resistance to attack may be limited. This is especially true when only MB or block is used as the basic shuffling unit, as shown in the left hand side image of Fig. 6 where some local structure can be somewhat guessed. In this case, since QP is relatively small, a video packet (with a pre-determined total number of bits) in many cases contains only a few MBs. As a result, the shuffling is restricted to local small regions, and the scrambled image is thus somewhat guessable if only MB or block is shuffled as a basic unit.

Shuffling run-level codewords will increase the shuffling space and further mess up the visual quality significantly. This can be seen from the right hand side image of Fig. 6 where the first 10 run-level codewords in each block are also shuffled (i.e., codewords in the index ranges of [0,0], [1,1], [2,2], …, [9, 9] are grouped and shuffled separately. DCT sign bits are encrypted instead of being shuffled, and are used for generating the shuffling tables). To further increase the scrambling effect and the resistance to attack, a larger clear-text unit such as a pre-determined number of rows of MBs should be used. Run-level codewords in different index ranges can also be mixed together to further increase the shuffling space. It is worthwhile to point out that both scrambled bitstreams in Fig. 6 are rendered using PV MPEG-4 player, without crash.

IV. ERROR CONCEALMENT BASED ATTACKS

Encrypted multimedia content is subject to error concealment based attacks. By error concealment based attacks, we mean attacks that are not based on cryptographic analysis, but rather, based on trying to conceal the resultant quality degradation by treating unbreakable encrypted data as lost and then attempting to minimize the impact on quality as a result of loss, using statistical information and knowledge of the media format. Such attacks are applicable to all selective encryption algorithms, regardless of format compliance. Such attacks, as a result of the systematically and intelligently selected fields that are subject to encryption (and therefore treated as lost in the attack), will usually result in significant loss of quality compared to the quality when the protected content is properly "unwrapped". For entertainment applications, such loss would usually render the content reconstructed by the attacker valueless. However, because of the separation of encoding and encryption in many applications, in some cases, the threat of error concealment based attacks is more severe than in other cases.

The effects of error concealment based attacks to prior art selective encryption methods have been studied in the literature, e.g. in [7], so in this section we would concentrate on the impact of error concealment based attacks on the methods proposed in this paper, still using MPEG-4 video as an example.

In our current implementations and simulations of the format-compliant selective encryption framework, we choose to encrypt motion vectors, Intra DC, DCT signs and Dquant (difference of quantization step-size QP between current and previous MB). Because of the nature of these fields, encrypting them will result in very scrambled looking sequences. However, because there are usually correlations between the values of these fields from neighboring regions in a frame, and all possible values of each individual field are not equally probable, and because of information leakage as a result of non-encrypted fields, one might use prior knowledge of the relative likelihood of field values, correlations between information in different packets, and other information to apply error concealment based attacks and attempt to obtain certain information from the encrypted video content. Therefore it is important to understand the impact of such attacks on various fields that are candidates for format compliant selective encryption, such as Dquant, DC and MVs.

- Dquant: Each Dquant value consists of only two bits, and is used to transmit the difference between the quantization parameter used for the corresponding MB and the previous MB. Not every MB has a Dquant field, as many MBs will use the default quantization parameter. Therefore, a simple and effective attack for encrypted Dquant is to set it to zero. In the figures presented in this paper, we used such attack for Dquant.

- INTRA DC: When intra DC values are encrypted, the resultant rendered video quality may look very scrambled. This is because the encrypted DC values are pretty much random numbers uniformly distributed over its range. Fig. 3 (Left image) shows a particular frame of the "soap" sequence with the intra

DC and DCT signs encrypted. The image is severely scrambled due to the rendered random color values. However, for most real images, DC values in neighboring 8x8 blocks exhibit high correlation. Thus one way to attack DC value encryption is to simply set all the DC values to a constant, e.g., to a constant DC value of 128 for luminance component and to 0 for chrominance components. As a result, the "effective" scrambling effect may not be as significant as the case where no attack is applied. Fig. 3 (Right image) shows the rendered image after such an attack. It can be seen that with such an attack, the background colors are no longer random and the contours of two people in the scene can be easily seen.

- MV: Motion Vectors in most video sequences also exhibit strong spatial correlation, both locally and sometime globally when there is panning. Furthermore, for most sequences, small MVs are much more likely than large MVs. Therefore, a simple attack is to simply set all MVs to 0. Fig. 4 (Right image) shows a rendered frame after such an attack. It can be seen that the visual quality is quite different with and without such an attack. In this example, because only MVs are encrypted and for this particular sequence, significant visual information is contained in texture information, setting MV to 0 is effective. However, for the same sequence, if we encrypt Intra DC and DCT signs in addition to MVs (Fig. 5) and apply the same attacks for all the encrypted fields (i.e. setting DC of luminance component to 128, DC of chrominance components to 0, and all MVs to 0), the resultant video sequence that the attacker obtains will be much less pleasant. This example demonstrates the importance of choosing the right combination of encryption/shuffling tools described in this paper for the particular type of content and application.

As discussed above, one problem with not encrypting DCT run-level codewords is that the spatial object contour of the scene can be comprehended by setting all MVs to zero and DCs to a constant. Combining shuffling with encryption will improve the robustness against error concealment based attacks, as it destroys some of the spatial correspondence between MBs or codewords within a shuffling group. Fig. 7 shows that the attack of simply setting DC of the luminance component to 128, DC of chrominance components to 0, and all MVs to 0 will not work for the proposed codeword shuffling scheme. However, because of the selective nature of shuffling, it is still subject to error concealment attacks if the attack is more intelligently designed and some complexity/delay is allowed. For example, instead of reducing visual scrambling effect of shuffling DC and MV information by setting the false MVs and DCs to most likely values, one can use the spatial/temporal smoothness constraint of the video frame to search for a good estimate of the video frame. Some fields such as intra-coded DCs and MVs are more vulnerable to such attack, although the attack is usually very expensive. To further enhance the security, these fields can be encrypted. In general, encrypting intra-coded DCs does not increase the visual scrambling effect much, compared to shuffling intra-DCs. However, it may help in some cases when the shuffling is too local, as shown in Fig. 8, where the face of the lady (in the upper-left quadrant) is somewhat visible when only shuffling is applied. This is mainly because shuffling is based on video packet as a *clear-text unit*, which in this case contains only a few local MBs with similar color.

In general, we observe that, error concealment based attacks for encrypted multimedia content are possible and in some cases effective and simple. This highlights the importance of designing the proper combinations of the encryption and shuffling tools to control the amount of information leakage and achieve the best trade off between security, complexity, delay, error resiliency, transcodability, and applicability of signal processing, etc.

For entertainment applications, we further observed that although error concealment based attacks may reduce the level of the scrambling effect achieved by the selective encryption and shuffling tools, the inevitable loss of quality as a result of such attacks in many cases still renders the reconstructed content valueless. For applications that have higher level of security requirement, such as nanny cam, security cam, sensitive video conferencing etc, more fields need to be encrypted or shuffled.

V. DISCUSSIONS

In summary, we described new methods of performing selective encryption and shuffling of compressed content bitstreams while preserving format compliance, and tested their performance with the MPEG-4 data partitioned error resilient mode syntax. We found that by selectively encrypting and shuffling various critical fields of the compressed bitstream, we could trade off complexity, security, bit rate overhead, and functionality. We also presented simulation results for securing MPEG-4 video content with some specific configurations of encrypted fields and cipher based on considerations of such tradeoffs.

Care needs to be taken when using the methods introduced in this paper and previous work in an encryption framework such as the MPEG-4 IPMP selective encryption framework [9]. As there is no one-size-fits-all solution, we could only provide the tools and the framework that facilitate the design of the best system for a given application scenario, instead of *the* best design itself. The tools, simulation and analysis will help one make the right choice in his/her own design. For wireless multimedia streaming to smart phones and PDAs where bandwidth and

computation are prime considerations, one can simply encrypt FLC fields, or encrypt a smaller portion of VLC coded fields and thereby reduce or avoid overhead. Due to the low original bandwidth, the "leak" of content related information is not critical, and the relative closed-system and low-powered nature of these devices add a hardware layer of protection. For high bandwidth, high quality, high value content, both FLC and VLC fields should be encrypted/shuffled to provide maximum scrambling of the content. Note that for such content, the bit rate overhead is relatively small.

Note that unlike encrypted data, encrypted multimedia content is subject to error concealment based attacks, which are not based on cryptographic analysis, but rather, based on trying to conceal the resultant quality degradation by treating unbreakable encrypted data as lost. Such attacks are common to all selective encryption algorithms and would usually still result in significant loss to quality. For entertainment applications, such loss may render the reconstructed content valueless. This brings one's attention to not just the possibility and effectiveness of error concealment based attacks, but also two points we have made in previous sections, i.e. the content protection mechanism has to be chosen based on many factors, not just complexity, or security alone; and secondly, it is important to have a normative configuration messaging framework such as that of MPEG-4 IPMP, so as to facilitate "on the fly" adjustment of content protection settings if individual modules in the system have been compromised, or the configuration of the system is deemed insufficient.

VI. REFERENCES

[1] W. Zeng and S. Lei, "Efficient frequency domain video scrambling for content access control," *Proc. ACM Multimedia'99*, pp. 285-294, Orlando, Nov. 1999. An extended version also to appear in *IEEE Trans. Multimedia*, 2002.

[2] J. Wen, M. Severa, W. Zeng, M. Luttrell and W. Jin, "A format compliant configurable encryption framework for access control of multimedia," in *IEEE Workshop on Multimedia Signal Processing*, pp. 435-440, Oct. 2001.

[3] J. Meyer and F. Gadegast, "Security mechanisms for multimedia-data with the example MPEG-1-video," *Project Description of SECMPEG*, Tech. Univ. of Berlin, Germany, 5/95.

[4] *ISO/IEC/SC29/WG11, "Information technology – Coding of audio-visual objects – Part 2: Visual ISO/IEC 14496-2"*, International Standards Organization, 11/98.

[5] L. Tang, "Methods for encrypting and decrypting MPEG video data efficiently," *Proceedings ACM Multimedia '96*, pp. 219-230, Boston, MA, 11/96.

[6] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm," *Proc. ACM Multimeida'98*, pp. 81-88, 1998.

[7] L. Qiao and K. Nahrstedt, "Comparison of MPEG encryption algorithms," *Inter. Journal on Computer & Graphics*, Special Issue on Data Security in Image Communication and Network, 22(3), 1998.

[8] A. S. Tosun and W. Feng, "Light-weight security mechanism for wireless video transmission," *Proc. IEEE Inter. Conf. on Information Technology: Coding and Computing*, pp. 157-161, April 2001.

[9] MPEG4 IPMP FPDAM, ISO/IEC 14496-1:2001/ AMD3, ISO/IEC JTC 1/SC 29/WG11 N4701, March 2002 .

[10] The Electronic Book Exchange System (EBX) Version 0.8. July 2000 Draft.

[11] *Data Encryption Standard (DES)*, FIPS PUB 46-3, Reaffirmed Oct. 25, 1999. Available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[12] *Announcing the Advanced Encryption Standard (AES)*, Nov. 26, 2001. FIPS-PUB 197, Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[13] R. Blom, E. Carrara, D. McGrew, M. Naslund, K. Norrman, D. Oran, "The Secure Real Time Transport Protocol (SRTP)", Internet Draft, Feb. 2001.

[14]I. Agi, L. Gong, "An empirical study of MPEG video transmission", *Proc. of the Internet Society Symposium on Network and Distributed Systems Security*, pp. 137-144, San Diego, CA, 1996.

[15]W. Zeng, J. Wen and M. Severa, "Fast self-synchronous content scrambling by spatially shuffling codewords of compressed bitstreams," to appear in *IEEE Inter. Conf. Image Proc.*, Rochester, Sept., 2002.

[16]P. Rogaway and D. Coppersmith, "A software-optimized encryption algorithm," *Fast Software Encryption*, Lecture Notes in Computer Science, Vol. 809, Springer-Verlag, pp. 56-63, 1994.

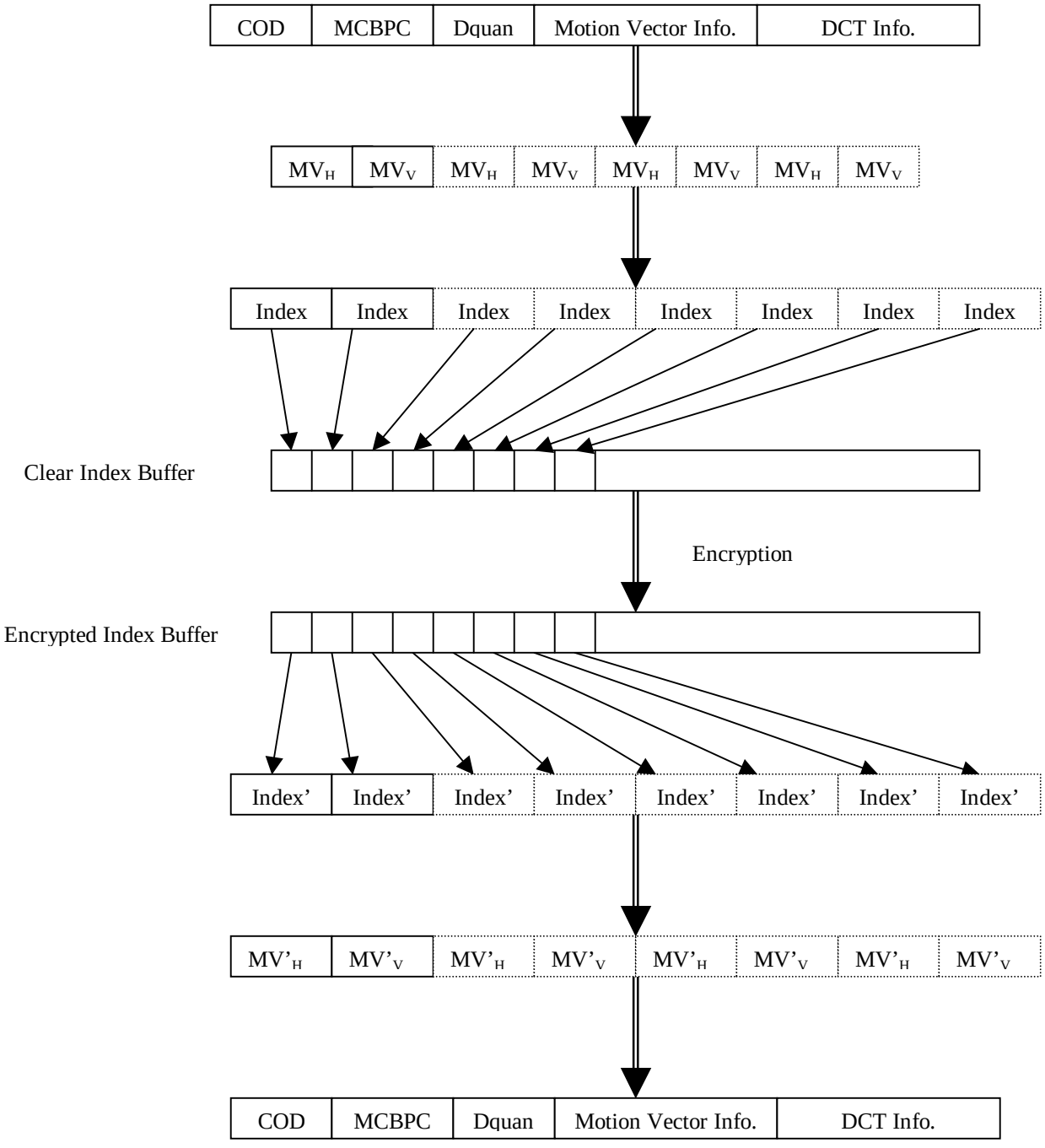

Figure 1: Encryption method for MV field

sequence	resolution	frame rate (fps)	bit rate (kbps)	overhead (MaxMV 8)	overhead (MaxMV 16)	overhead (MaxMV 32)
soap city	160x112	10	263	< 1%	1%	2%
	160x112	10	71	4%	8%	11%
	160x112	5	136	< 1%	1%	2%
	160x112	5	18	6%	13%	19%
hanging up	320x224	10	158	7%	10%	17%
	320x224	10	86	10%	21%	30%
	320x224	10	531	2%	3%	5%
head	176x144	10	29	8%	13%	18%
	176x144	10	17	10%	16%	22%
	176x144	5	26	8%	13%	17%
	176x144	5	5	17%	29%	36%

Table 1: Overhead for MVD encryption. "MaxMV N" indicates that only MVDs within the range of [-*N*, *N*] (measured in half-pel) are encrypted

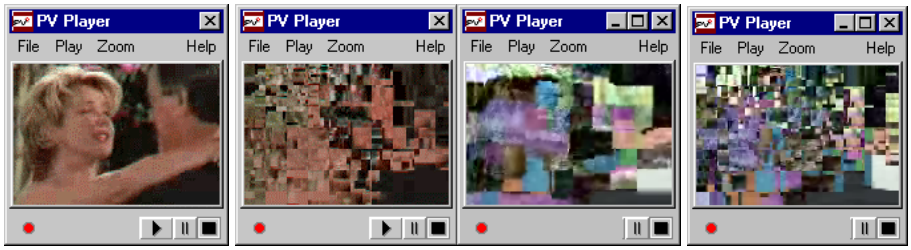

Figure 2: Example sequence (from Left to Right): Original, Encrypt VLC only, Encrypt FLC, and Encrypt VLC&FLC.

Fig. 3: One encrypted frame of "soap" sequence (encoded at 384 kbps). Left: only Intra DCs and DCT signs are encrypted; Right: resultant image after being attacked by setting DCs of luminance component to 128 and DCs of chrominance components to 0.

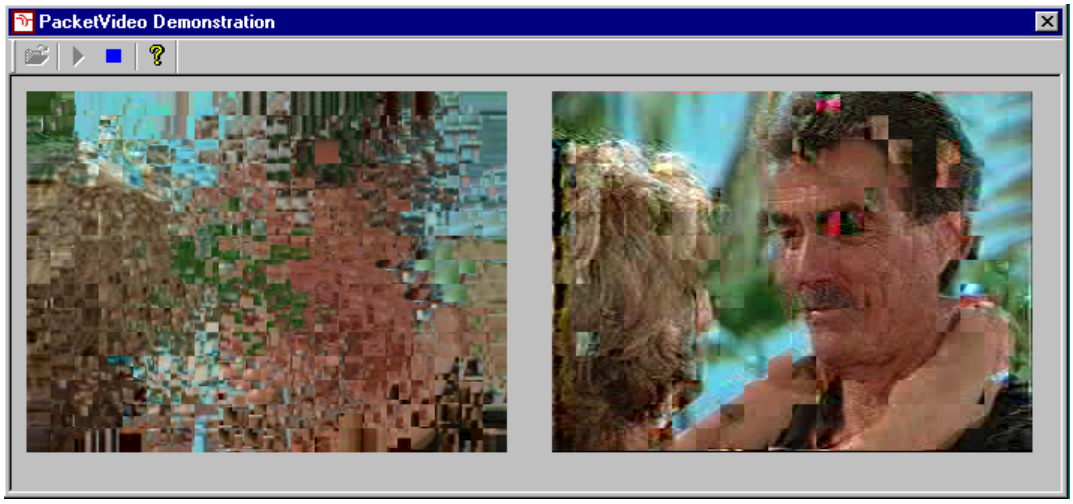

Fig. 4: One encrypted frame of "soap" sequence (encoded at 384 kbps). Left: only MVs are encrypted; Right: resultant image after being attacked by setting all MVs to 0.

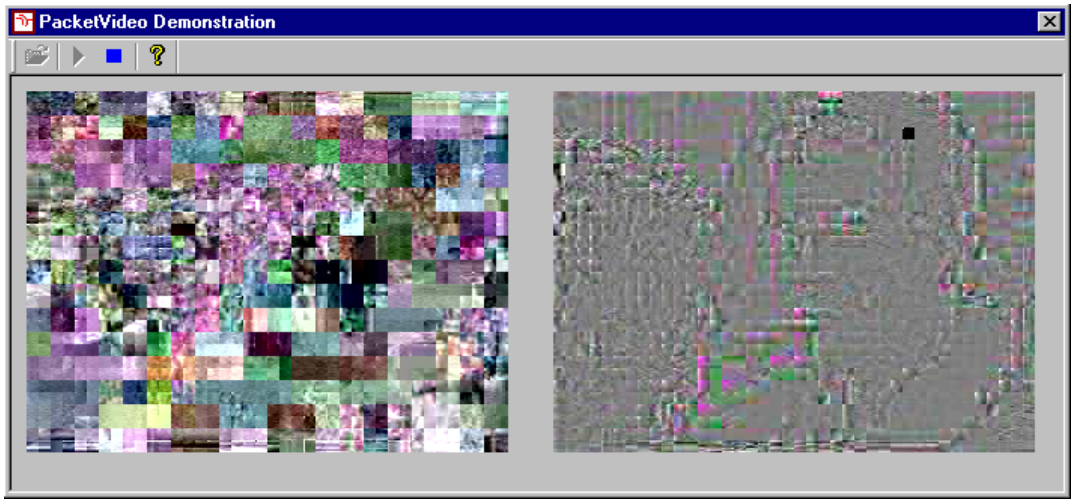

Fig. 5: One encrypted frame of "soap" sequence (encoded at 384 kbps). Left: Intra DCs, DCT signs and MVs are encrypted; Right: resultant image after being attacked by setting DCs of luminance component to 128, DCs of chrominance components to 0, and all MVs to 0.

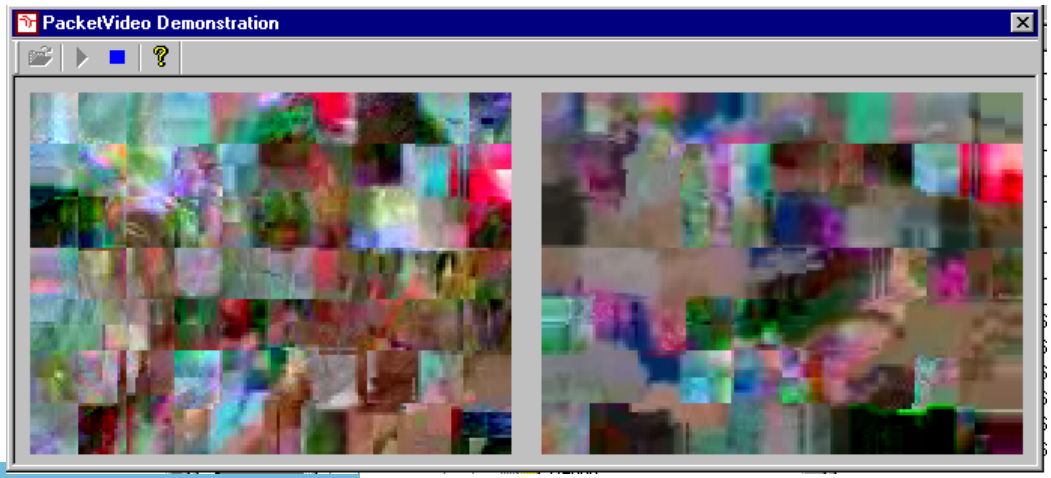

Fig. 6: One scrambled frame of "soap" sequence (5fps with const QP of 4), shuffling is based on video packet as a *clear-text unit*. Left: only MBs and 8x8 blocks are shuffled (using different shuffling tables); Right: MBs, 8x8 blocks, and the first 10 run-level codewords in each block are shuffled separately.

Fig. 7: Attacked frame of Fig. 6 by setting DCs of luminance component to 128, DCs of chrominance components to 0, and all MVs to 0.

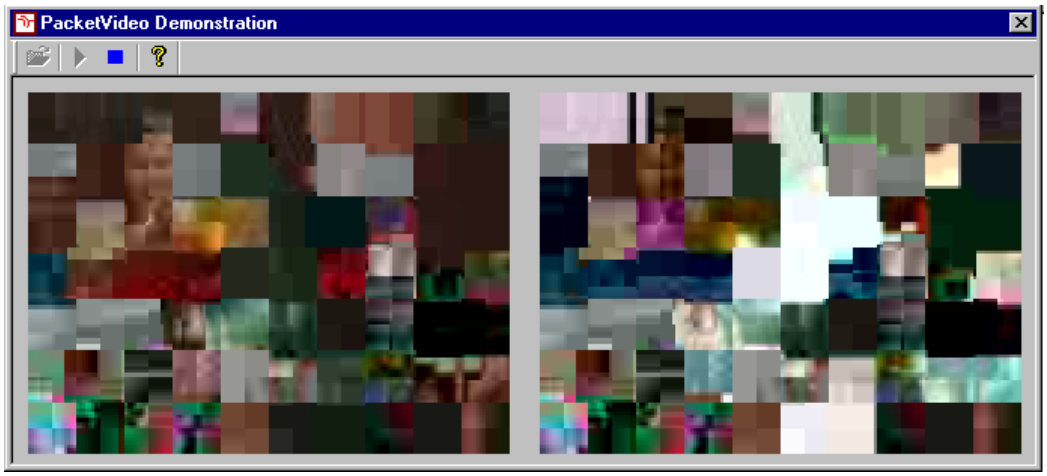

Fig. 8: One scrambled frame of "soap" sequence (5fps with const QP of 24), shuffling is based on video packet as a *clear-text unit*. Left: MBs, 8x8 blocks, and the first 5 run-level codewords in each block are shuffled separately; Right: MBs, 8x8 blocks, the first 5 run-level codewords in each block are shuffled separately, intra-DCs are encrypted as well.