# FORMAT INDEPENDENT ENCRYPTION OF GENERALIZED SCALABLE BIT-STREAMS ENABLING ARBITRARY SECURE ADAPTATIONS

*Debargha Mukherjee*[*], *Huisheng Wang*[+], *Amir Said*[*], *Sam Liu*[*]
[*] *Hewlett Packard Laboratories, Palo Alto, CA*
[+] *Dept. of Electrical Engineering, University of Southern California, CA*

## ABSTRACT

Secure format-independent adaptation of bit-streams during delivery is becoming increasingly important to cope with content piracy while still accommodating diverse networks, terminals and formats. Generalized scalable bit-streams are particularly advantageous in this regard, since they enable a variety of efficient and secure adaptations. Further, by associating such a bit-stream with appropriate metadata, such as those standardized in MPEG-21 Part 7 entitled Digital Item Adaptation (DIA), the adaptation process can be fully format-independent. In this paper, to maximally secure a generalized scalable bit-stream while allowing arbitrary encrypted domain adaptations, strong progressive encryption methods are extended to multiple dimensions. Further it is shown that by appropriate modeling of such bit-streams and re-use of some DIA descriptions, the encryption and decryption engines themselves can be entirely metadata-driven and format-independent. This leads to end-to-end format-independent secure and adaptive delivery architectures for scalable bit-streams.

## 1. INTRODUCTION

To improve multimedia content accessibility and to maximize experience commensurate with diverse and dynamic terminal and network capabilities and conditions, as well as individual preferences, it is essential to adapt multimedia content in the delivery path to end consumers. At the same time, unauthorized access to the content during distribution must be prevented to protect the rights of legitimate content owners. The solution is to use encryption methods that allow possibly untrusted adaptation engines in the delivery path to adapt content in the encrypted domain. A third factor is that the set of rich media content formats to be delivered is growing fast. This justifies a drive towards delivery infrastructure components, such as adaptation or encryption/decryption engines or modules thereof that use a universal processing model – which do not need frequent upgrades to support new formats and can even support proprietary ones. This is enabled by associating the content with standardized metadata that is small enough to make delivery alongside the content feasible, and not detailed enough to leak information about the content from a security stand-point.

To satisfy the triple needs for adaptability, security and format-independence during delivery, *scalable* bit-streams [1][2] appear promising. From the progressive dependencies of a scalable bit-stream it follows that secure adaptation is possible by use of progressive encryption techniques, as shown in [3][4][5]. However, because scalable bit-streams frequently contain multiple dimensions of scalability, it is necessary to extend progressive encryption to handle multi-dimensional dependencies.

In prior work [6][7][8], format-independent metadata-driven adaptation of generalized scalable bit-streams was reported. Recently, such metadata has been standardized in Part 7 of the emerging MPEG-21 standard [9] entitled Digital Item Adaptation (DIA) [10]. In this work, we show that some of the same metadata enables the encryption/decryption processes to be format-independent as well. In combination this yields secure and adaptive delivery architectures for scalable bit-streams, where the operation of all encryption, adaptation and decryption engines in the delivery path rely solely on incoming metadata and not on pre-knowledge of the format.

Fig. 1 shows a conceptual model for such a delivery architecture. A scalable bit-stream originating from a server is encrypted by an encryption engine based on associated metadata and a key provided by a license server. The encrypted content and the unencrypted metadata is then transmitted over an insecure channel, where both are adapted possibly multiple times, by adaptation engines based on adaptation constraints specified as in [6][7][8][10] derived from network and terminal capabilities, conditions and preferences. Finally, the decryption engine decrypts the encrypted content based on currently associated metadata and recipient's key, and delivers a compliant bit-stream to an authorized recipient. Note that Fig. 1 does not show initialization vectors (IV) that many encryption schemes use. The IVs are typically generated randomly and conveyed to the decryption engine in the clear. In Fig. 1, they would be included in the metadata output from the encryption engine.

In the rest of the paper, we first present adaptation models for fully scalable bit-streams, and then derive the principles and present some instances of multi-dimensional progressive encryption of such bit-streams. Finally, an implementation of encryption and decryption engines using MPEG-21 DIA descriptions is reported.

## 2. MODELING SCALABILITY

For a scalable bit-stream, the bulk of the adaptation task is deletion of bit-stream segments, which is often followed by update operations on a few bit-stream fields as required for format compliance. In this section, the delete and update operations are modeled and examined.

### 2.1. Delete

The inherent dependencies in a generalized scalable bit-stream, lead to a universal logical model for all scalable bit-streams, referred to as

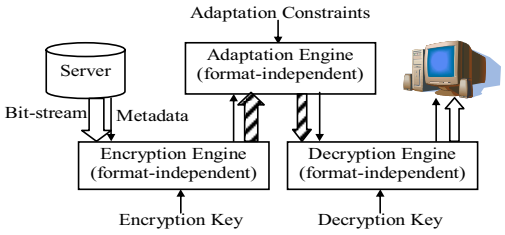
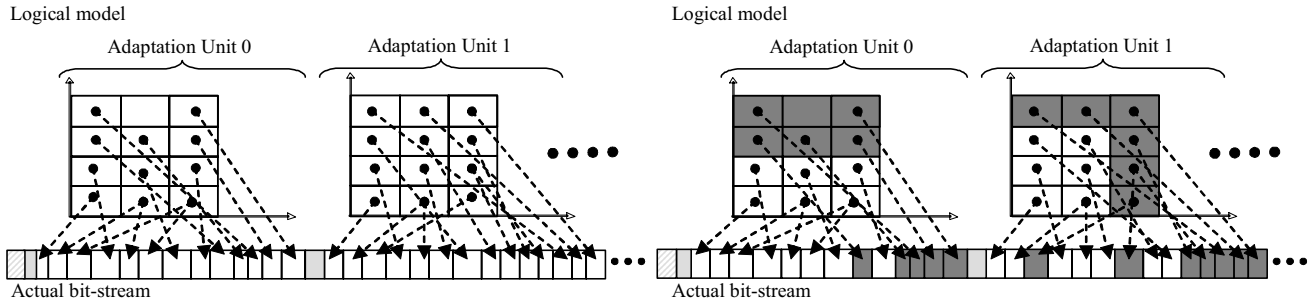



Fig. 1 Format-independent delivery architecture.




(a) Pre-adaptation logical model and actual bit-stream

(b) Post-adaptation logical model and actual bit-stream

Fig. 2 Bit-stream example showing two adaptation units: (a) with 3x4 logical hypercubes each, (b) with 3x2 and 2x3 logical hypercubes, after adaptation of the bit-stream in (a). Dark-shaded logical units and segments are assumed deleted in (b). Light-shaded and cross-hatched segments correspond respectively to update fields and segments that are neither included in logical units nor are update fields.

the SSM model, proposed in [6][7][8]. This model constrains how segments are deleted during adaptation.

According to a simplified version of this model, the data in an elemental scalable bit-stream is organized in sequentially processed units called *adaptation units*, on which adaptation decisions are made and applied. Examples of adaptation units are GOP, Frame, Tile, etc. Further, within each adaptation unit, the data is organized in a logical hypercube, with a variable number of dimensions. If there are $L$ dimensions of scalability in each adaptation unit, and the $i$th dimension of the $n$th adaptation unit contains $l_i[n]$ layers, we can say that the data in the $n$th adaptation unit consists of $l_0[n]\times l_1[n]\times\ldots\times l_{L-1}[n]$ logical data segments $B(j_0, j_1, \ldots, j_{L-1})[n]$, $j_i$=0,1,…, $l_i[n]$–1, $i$=0,1,…, L–1, called *logical units*, arranged in a hypercube. A logical unit may consist of any number of contiguous bit-stream segments located arbitrarily in the actual bit-stream. Fig. 2 (a) shows the logical structure and mapping to the actual bit-stream for two adaptation units of a bit-stream.

From the logical structure, generally speaking any arbitrary selection of layers can be deleted for each dimension, during adaptation. For simplicity we only consider fully scalable bit-streams, where layers can only be dropped from the outer ends for each dimension. Fig. 2(b) shows an adaptation of the bit-stream in Fig. 2(a), where 3x2 and 2x3 logical hypercubes from the origin are kept for two adaptation units, the rest being deleted. Correspondingly, bit-stream segments that the deleted logical units map to are deleted.

From the above model for deletion, it follows that *causality* of the logical units must be maintained during encoding/encryption, so that a decoder/decrypter can handle adapted content. That is, to encode/encrypt logical unit $B(j_0, j_1, \ldots, j_{L-1})[n]$, the encoder/encrypter can only use information from logical units $B(k_0, k_1, \ldots, k_{L-1})[n]$, where $k_i \le j_i$ for all $i$, and $k_i \ne j_i$ for at least one $i$. This would ensure that for any valid adaptation, the decoder/decrypter can still decode/decrypt the content unambiguously.

## 2.2. Update

Often in scalable bit-streams it is not just enough to delete logical units, but also perform other update operations on certain fields in the bit-stream, to ensure format-compliance. These fields are often of a fixed-length. For instance, there may be fields conveying information such as image/frame dimensions, number of layers included and so on, that need to be updated in the bit-stream after deletion of logical units. There may also be fields that specify lengths of certain bit-stream segments or locations of other parts of the bit-stream that are likely to need updates when logical units are deleted. Further, there may be fields representing sequential counters, such as packet number or frame number (temporal reference) that also are likely to need updates when logical units are deleted.

While the exact mechanism for obtaining correct updates is irrelevant for encryption, it is essential that these fields be left unencrypted, so that adaptation engines in the delivery path can modify these fields without affecting the decryption of the rest of the bit-stream. Fortunately, for most fully scalable bit-streams, these fields constitute a minor portion of the bit-stream, so that security breach is minimal. The light-shaded parts of the bit-stream in Fig. 2(a) and (b) represent some update fields that are left unencrypted.

## 3. ENCRYPTION SYSTEM

The first step in encrypting a generalized scalable bit-stream is to obtain the *encryptable* plaintext with a known length in bits for each logical unit, by concatenating bit-stream segments corresponding to the logical unit while excluding updateable fields. The mapping from logical co-ordinates to corresponding bit-stream segments as well as the location of the update fields, required for this step is conveyed by means of a high-level syntax description metadata associated with the bit-stream. The encryptable plaintext for each logical unit is encrypted by a Logical Unit Encrypter (LUE) yielding a ciphertext having the same length in bits as the plaintext, which is then substituted into the bit-stream to create the secure bit-stream.

While in principle, each logical unit can be encrypted independently, to ensure maximal security they must be encrypted in a multi-dimensional progressive manner, so that encryption and/or decryption of a given logical unit become impossible without encryption and/or decryption of logical units causal to it. Assume the LUE is associated with an initial state $S_{in}$, and on encrypting a plaintext P of specified length reaches an end state $S_{out}$, where in general $S_{out} = f(S_{in}, P)$, $f(.)$ being an arbitrary function. Then, progressiveness can be ensured by making the initial state $S_{in}$ of the LUE for a given logical unit depend on the end states $S_{out}$ of the LUE for previous logical units along each dimension. If $S_{in}(j_0, j_1,\ldots, j_{L-1})[n]$ and $S_{out}(j_0, j_1,\ldots, j_{L-1})[n]$ denote binary initial and ending states of the LUE for the $(j_0, j_1,\ldots, j_{L-1})$th logical unit plaintext for the $n$th adaptation unit, then one can use:

$$S_{in}(0, 0,\ldots, 0)[n] = I_V[n]$$

$$S_{in}(j_0, j_1,\ldots, j_{L-1})[n] = S_{out}(j_0-1, j_1,\ldots, j_{L-1})[n] \oplus S_{out}(j_0, j_1-1,\ldots, j_{L-1})[n] \oplus\ldots\oplus S_{out}(j_0, j_1,\ldots, j_{L-1}-1)[n],\ j_i=0,1,\ldots, l_i[n]-1.$$

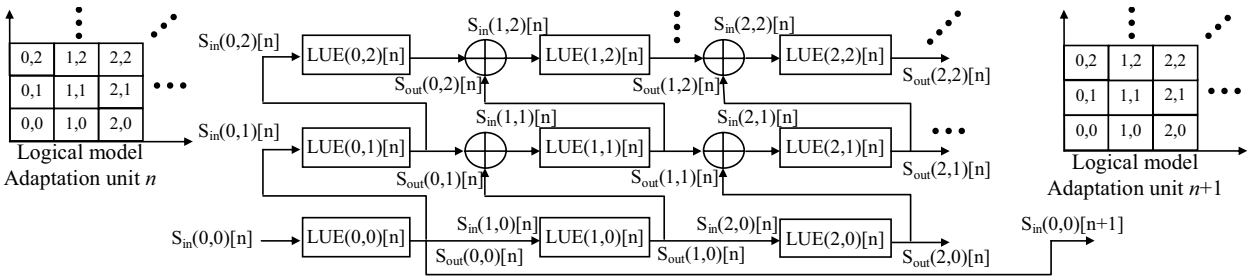



Fig. 3 Multi-dimensional state propagating encryption for generalized scalable bit-streams. A 2-dim example is shown. LUE($i$,$j$)[$n$] is the Logical Unit Encrypter for the ($i$,$j$)th logical unit of adaptation unit $n$. It is implicit here that each LUE takes in a logical unit plaintext having a specified length and yields the corresponding ciphertext based on a key.

It is assumed that $S_{out}(j_0, j_1,\ldots, j_{L-1})[n] = 0$, when any $j_i<0$. $I_V[n]$ is an initialization vector for the $n$th adaptation unit, typically generated randomly by an encrypter and sent in the clear to the decrypter. Alternatively, it can be generated by a counter synchronized at the decrypter. In many cases, since the logical unit with all zero co-ordinates can always be assumed to be transmitted, it is possible to further propagate the states along adaptation units, using:

$$S_{in}(0, 0,\ldots, 0)[n+1] = I_V[n+1] = S_{out}(0, 0,\ldots, 0)[n],$$

with periodic resynchronization with a newly generated IV at intervals of a group of adaptation units. Fig. 3 shows the above state propagation mechanism for a 2-dimensional logical model.

Note that in a scalable bit-stream because the critical data is usually in the initial layers, to reduce complexity only a partial set of logical units $B(j_0, j_1, \ldots, j_{L-1})[n]$ with $j_i$=0,1,…, $\min(m_i, l_i[n])$–1 can be encrypted and the rest left unencrypted. Here $m_i$ is the max number of layers to be encrypted for the $i$th dimension of each adaptation unit.

For the design of the actual LUE, block ciphers operated in one of a variety of *modes* is considered. Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output feedback (OFB), and Counter are some common modes of operation with several variants available for each. But for the strongest possible encryption we seek modes that are infinitely *error propagating*. In such modes, not only is the encryption operation progressive, but also any change in a ciphertext block makes it impossible to decrypt correctly the current and all future blocks. It has been shown based on cryptographic arguments [12] that such modes can effectively prevent birthday attacks and diffuse patterns in the plaintext better. The traditional modes, CBC for example, does not satisfy this property. Further, error propagating modes are better matched to the progressive decoding dependencies that already exist in a generalized scalable bit-stream.

In particular, we use a recently proposed family of error propagating modes – called Accumulated Block Chaining (ABC) [12], with roots in Campbell's Infinite Garble Extension (IGE) mode [13]. In order to ensure that the ciphertext has the same length in bits as the plaintext, a method called ciphertext stealing originally proposed in [14] is adopted at the end of the logical unit. A LUE based on ABC with ciphertext stealing is shown in Fig. 4. The initial state is given by $S_{in} = \{H_{-1}, C_{-1}\}$. Let the plaintext be $(P_0,P_1,\ldots,P_{n-1}, P_n)$, where $P_0$ through $P_{n-1}$ are full blocks (64 or 128 bits depending on the size of the block cipher $E_k$) and $P_n$ be a final short block. Let the corresponding ciphertext be $(C_0, C_1,\ldots, C_{n-1}, C_n)$, where $C_n$ is short. The encryption steps are:

$H_i = h(H_{i-1})\oplus P_i$, $C_i=E_k(C_{i-1}\oplus H_i) \oplus H_{i-1}$, $i$=0,1,…,$n$–2

$H_{n-1} = h(H_{n-2})\oplus P_{n-1}$, $C_n \| C' = E_k(C_{n-2}\oplus H_{n-1}) \oplus H_{n-2}$

$H_n \| H' = h(H_{n-1})\oplus P_n \| 0$, $C_{n-1}=E_k(C_n \| C' \oplus H_n \| P') \oplus H_{n-2}$

h(.) is a simple function, ex. h(X)=X or h(X)=X>>1 (>> denoting circular shift). The case h(X)=0 corresponds to the IGE mode [13].

The last two lines of the above equations implement ciphertext stealing [14]. Here ‖ denotes concatenation of two short blocks to obtain a full sized block. The first has the same length as $P_n$. P' is a padding pattern known to both encrypter and decrypter based on the length of $P_n$. C' and H' are not transmitted. The decryption steps are:

$H_i = D_k(H_{i-1}\oplus C_i) \oplus C_{i-1}$, $P_i = h(H_{i-1})\oplus H_i$, $i$=0,1,…,$n$–2

$H_n \| C' = D_k(C_{n-1}) \oplus C_n \| P'$, $H_{n-1} = D_k(H_{n-2}\oplus C_n \| C') \oplus C_{n-2}$

$P_{n-1} = h(H_{n-2})\oplus H_{n-1}$, $P_n \| H' = h(H_{n-1})\oplus H_n \| 0$

The ending state of the LUE propagated for encrypting/decrypting successive logical units in each dimension is: $S_{out} = \{H_{n-1}, C_{n-1}\}$.

Although we show explicitly only one mode that arguably provides the strongest encryption for a generalized scalable bit-stream, any of the simpler modes (CBC etc.) or variations thereof can be used just as well, with appropriate improvisations to handle initial and end states.

Instead of block ciphers, stream ciphers can be used for the LUE as well. In this case, the internal state of the pseudo-random key-stream generator is initialized with $S_{in}$, and at the end of encryption of a logical unit, the end interrnal state is propagated forward as $S_{out}$. Multi-dimensional state propagation still works as in Fig. 3.

## 4. IMPLEMENTATION USING MPEG-21 DIA

In this section, we present an actual implementation of a format-independent encryption/decryption engine driven by metadata standardized recently under MPEG-21 DIA.

To encrypt a generalized scalable bit-stream to allow arbitrary adaptations, the only information that the encrypter needs are the model parameters, the locations of the actual bit-stream segments that each logical unit map to, and the locations of the updateable fields. Because this is a subset of the information needed for adaptation of a scalable bit-stream, it is obvious that some of the DIA descriptors can be re-used for encryption. A DIA description called the Bit-stream Syntax Description (BSD) and its variant the generic BSD (gBSD) provides the high-level syntax of a bit-stream. The exact adaptation operation is modeled and conveyed by means of an XML transformation applied to the (g)BSD. While the XML transformation language is non-normative in DIA, a model based transformation entitled BSD Transformation Instructions (BSDTrI) was proposed [15][16] for scalable bit-streams during the standard development process that explicitly provided Delete and Update Instructions as in Section 2. The gBSD and the BSDTrI together convey the information necessary for a format-independent encryption/decryption engine to encrypt/decrypt a generalized scalable bit-stream in a multi-dimensional progressive manner.

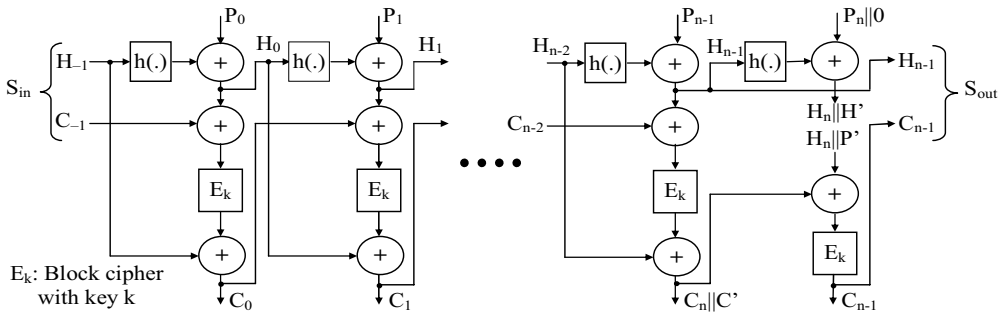



Fig. 4 LUE based on Accumulated Block Chaining (ABC) mode. When h(X)=0, ABC becomes Infinite Garble Extension (IGE) mode.

The overall encryption-adaptation-decryption chain based on DIA descriptions is shown in Fig. 5. The bit-stream originating from a server is associated with a variety of metadata to enable encryption and adaptation. Among them, the ones processed by the Encryption engine are the gBSD and the BSDTrI. Note that because the encryption is length preserving, the gBSD and the BSDTrI remain the same for both un-encrypted and encrypted content. The gBSD and the BSDTrI along with other DIA descriptions [10], for instance AdaptationQoS (AQoS), Universal Constraints Description (UCD) and Usage Environment Descriptions (UED), are next processed by a mid-stream format-independent adaptation engine [8] to yield an adapted but encrypted bit-stream as well as the corresponding adapted gBSD, denoted gBSD'. gBSD' and BSDTrI are processed by a decryption engine to yield an adapted clear bit-stream.

The gBSD and BSDTrI driven encryption/decryption engine developed use the 128 bit AES [17] block cipher with a 128 bit key in ABC/IGE modes. The encryption-adaptation-decryption chain was tested on various scalable formats, including: (a) MPEG4 Visual Elementary Stream with temporal scalability using B-VOPs in a 1-dim scalability structure; (b) MPEG4 Visual Texture Coding with spatial and SNR scalability in a 2-dim scalability structure; (c) JPEG2000 [1] in various progression modes, having spatial, SNR and color scalability in a 3-dim scalability structure. (Precinct or Tile-based ROI scalability is not covered in this paper); (d) MC-EZBC [2] – a fully scalable video format having simultaneous temporal, spatial and SNR scalability in a 3-dim scalability structure.

All these use cases were tested for format-independent adaptation during the course of development of the DIA standard. The JPEG2000 and MC-EZBC use cases have been reported in [8]. In the current work, the adaptation operation is preceded by the encryption, and followed by decryption operation. The final adapted bit-stream after decryption was found to be correctly decodable in all cases.

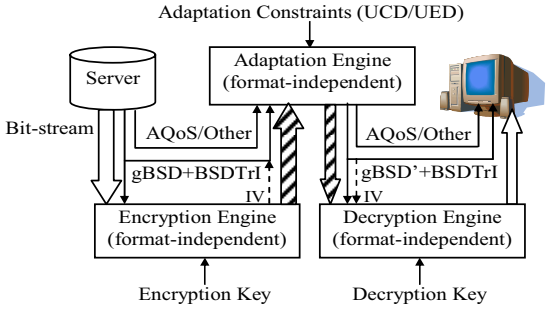



Fig. 5 Format-Independent Encryption-Adaptation-Decryption chain based on MPEG-21 DIA descriptions. Initialization vectors (IV) that need to be sent to the decryption engine can be integrated into the gBSD during encryption.

## 5. REFERENCES


[1] D. S. Taubman, M.W. Marcellin, "JPEG2000: Image Compression Fundamentals, Standards and Practice," *Kluwer Acad. Pubs, 2002*.

[2] S. –T. Hsiang, J. W. Woods, "Embedded video coding using invertible motion compensated 3-D subband/wavelet filter bank," *Signal Processing: Image Communication*, vol. 16, pp. 705-24, May 2001.

[3] S. J. Wee, J. G. Apostolopoulos, "Secure scalable video streaming for wireless networks," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, vol. 4, pp. 2049–2052, May 2001.

[4] S. J. Wee, J. G. Apostolopoulos, "Secure scalable streaming enabling transcoding without decryption," *Proc. IEEE Int. Conf. Image Processing,* vol. 1, pp. 437–440, Oct. 2001.

[5] S. J. Wee, J. G. Apostolopoulos, "Secure scalable streaming and secure transcoding with JPEG-2000," *Proc. IEEE Int. Conf. Image Processing,* Sept. 2003.

[6] D. Mukherjee and A. Said, "Structured Scalable Meta-formats (SSM) for Digital Item Adaptation," *Proc. SPIE, Internet Imaging IV*, vol. 5018, pp. 148-67, Jan 2003.

[7] D. Mukherjee, P. Chen, S-T. Hsiang, J. Woods, and A. Said, "Fully Scalable Video Transmission using the SSM Adaptation Framework," *Proc. SPIE, Visual Commun. and Image Proc.*, vol. 5150, July 2003.

[8] D. Mukherjee, G. Kuo, S. –T. Hsiang, S. Liu, A. Said, "Format-independent scalable bit-stream adaptation using MPEG-21 DIA," *Proc. IEEE Int. Conf. Image Processing*, Singapore, Oct 2004.

[9] J. Bormans, J. Gelissen, A. Perkis, "MPEG-21: The 21st century multimedia framework", *Signal Processing Mag., IEEE*, Volume 20, Issue 2, pp. 53 – 62, March 2003.

[10] "ISO/IEC 21000-7 FDIS Part 7: Digital Item Adaptation," *ISO/IEC JTC 1/SC 29/WG 11/N6168*, Dec 2003, Hawaii, USA.

[11] G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers, M. Amielh, "Bitstream syntax description: A tool for multimedia resource adaptation within MPEG-21," *Signal Processing: Image Communication, Special Issue on Multimedia Adaptation*, vol. 18, pp. 721-747, Sep 2003.

[12] L. R. Knudsen, "Block chaining modes of operation," *Report in Informatics* No. 207, Dept. of Informatics, University of Bergen, Norway, Oct 2000.

[13] C. Campbell, "Design and Specification of Cryptographic capabilities," *D. Barnstad, Ed., National Bureau of Standards Special Pub.*, US Dept. of Comm., pp. 54-66, Feb 1978.

[14] J. Daeman, "Cipher and hash function design," *Ph.D. thesis*, Katholieke Universiteit Leuven, Mar 1995.

[15] D. Mukherjee, G. Kuo, S. Liu, G. Beretta "Model-based lightweight BSD Transformation Instructions and Reserved Marker Token Formats," ISO/IEC JTC1/SC29/WG11 MPEG2003/M9772, July 2003.

[16] D. Mukherjee, H. Wang, S. Liu, "On BSD Transformation Instructions – Streaming implementation and updates," ISO/IEC JTC1/SC29/WG11 MPEG2004/11091, July2004.

[17] "Specification for the Advanced Encryption Standard (AES)," *Federal Information Processing Standards*, Publication 197, Nov 2001.