Adaptation-aware encryption of scalable H.264/AVC video for content security

H. Kodikara Arachchi[a,*], X. Perramon[b], S. Dogan[a], A. M. Kondoz[a]

[a] I-Lab, Centre for Communication Systems Research (CCSR), University of Surrey, Guildford GU2 7XH, Surrey, UK
{H.KodikaraArachchi, S.Dogan, A.Kondoz}@surrey.ac.uk

[b] Dept. of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain
xavier.perramon@upf.edu

ABSTRACT
Data encryption is one of the key information security technologies used for safeguarding multimedia content from unauthorised access and manipulation in end-to-end delivery and access chains. This technology, combined with the appropriate cryptographic methods, effectively prevents the content against malicious attacks, so as to protect its authenticity as well as integrity. While encryption based security is ensuring the authorised consumption of the multimedia content, content adaptation technologies have the primary goal of providing means for wider dissemination of the content across diverse networks, devices and users, and thus enriching user satisfaction and experience of the delivered content within a given set of usage environment constraints. Traditionally, protected contents can only be adapted at trusted adaptation engines residing between the source and end-users since they have to be fully decrypted before performing the necessary adaptation operations. The drawback of such a process is that it significantly limits the availability and flexibility of adaptation engines applicable for adapting protected contents on the fly. Thus, this paper proposes a novel scalable H.264/Advance Video Coding (AVC)-compatible video encryption technique, which is also transparent to adaptation engines in an end-to-end video delivery scenario. The proposed technology relies on keeping syntax elements required for performing the adaptation operations clear (i.e., not encrypted). The effectiveness of the proposed technique has been successfully verified in scenarios, where both conventional Joint Scalable Video Model (JSVM) bit stream extracting and random packet dropping mechanisms are used.

* Corresponding author. Tel: +441483684742, Fax: +441483686011

1. Introduction

Today's multimedia communication landscape has been greatly shaped by the coexistence of a number of complementary as well as competing codec, access, delivery and consumption technologies. With this heterogeneity of the underlying technologies in mind, guaranteeing the Quality of Experience (QoE) [1][2] expected by users is a nontrivial exercise. In addition to the technological factors, the diversity of user preferences, as well as when and where the content is consumed add additional dimensions to the already complex dilemma. As a result, under the umbrella of the Universal Multimedia Access (UMA) concept, the notion of transparent access to rich multimedia content is widely discussed in the research community [3]. The MPEG-21 standard, one of the most recognised efforts to pave the way to the success of UMA, has promoted the content adaptation to achieve the goals of the UMA [4].

While UMA is laying out the foundation for seamless access to multimedia resources, there is also an undeniable demand for certain restrictions to accessing protected multimedia contents. In the wake of a technological challenge to prevent sheer levels of piracy, the entertainment industry well understood the importance of such restrictions and used Digital Rights Management (DRM) technologies to protect their invaluable contents. These techniques effectively limit the use of copyrighted contents. It is not only the consumer multimedia industry that takes measures to prevent the unauthorised access of the content, but also those who have a pressing need for protecting sensitive contents delivered over hard-to-trust communication infrastructures such as users of Virtual Collaboration Systems (VCS) also benefit from these technologies. However, these content security technologies also limit the content adaptation possibilities.

This paper focuses on protecting visual media content through encryption, in order to prevent unauthorised access. Traditionally, protected content can only be adapted at trusted adaptation engines residing between the source and end-users since such content has to be fully decrypted before performing the necessary adaptation operations. Therefore, only a trusted set of Adaptation Engines (AEs) can be used for adapting protected contents. However, this restriction effectively limits the choice of AEs. To the best of our knowledge, any attempt to eliminate the need of a trusted AE for adapting encrypted scalable video has not been reported in the literature to date. Video adaptation and encryption has been discussed together for encrypting adapted video contents only. The techniques proposed in [5],[6],[7] and [8] assume that either the content is received unencrypted or they are decrypted prior to performing the adaptation operation. Consequently, none of these techniques are fit for end-to-end adaptation architectures. Thus, this paper proposes a novel adaptation-aware encryption concept for securing scalable video. The paper also proposes enabling technologies for Scalable Video Coding (SVC) extension of H.264/AVC-compatible (H.264/SVC) [9][10] video to achieve transparency to AEs in an end-to-end delivery scenario. The high level view of such a scenario, in which the proposed adaptation-aware encryption architecture is employed, is illustrated in Figure 1.The proposed encryption mechanism offers full protection to scalable content while still providing necessary transparency for AEs to perform adaptation operations. The transparency is achieved by encrypting only a part of the bit stream so that necessary syntax elements needed for performing intended adaptation operations are available without decrypting the content. One of the advantages of the proposed algorithm is that a compatible decryptor is capable of identifying the encrypted segments even without any assistance from the encryptor through signalling. Another advantage of the proposed technique is that it can identically be used in both the packet oriented and byte stream [10] oriented transmission scenarios.

The rest of the paper is organised as follows: Section 2 briefly introduces the background on the encryption and scalable video adaptation concepts. Section 3 presents the security architectures for content adaptation followed by the description of the proposed adaptation-aware encryption technique in Section 4. Section 5 discusses experimental results, and finally Section 6 concludes the paper.

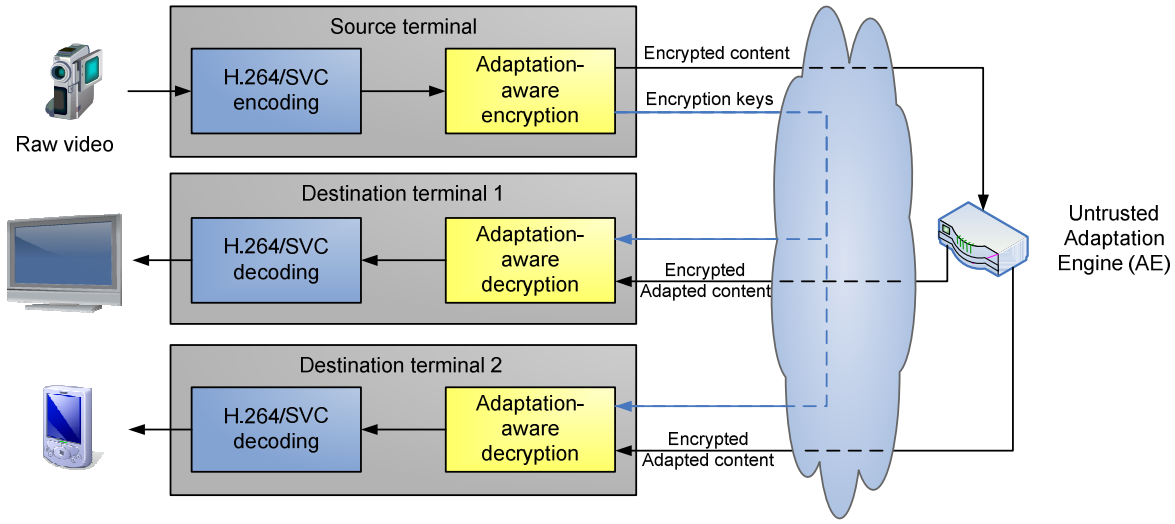


Figure 1. High level view of a typical end-to-end secure content delivery scenario

2. Background

2.1. Introduction to basic cryptographic algorithms

This subsection introduces basic concepts related to symmetric cryptographic algorithms, i.e., those in which the same key is used for encryption and decryption. Further details on these concepts can be found in various references, such as [11], [12] and [13].

Two basic types of symmetric encryption algorithms exist: stream ciphers and block ciphers. Stream cipher combines the plaintext with a pseudo-random bit sequence, known as keystream, generated from the encryption key. In contrast, the block cipher encryption algorithm works on separate input blocks of fixed length, typically 64 or 128 bits, to produce output blocks of the same length. Encryption with a block cipher may require padding of the input plaintext in order to process an integral number of blocks.

Stream ciphers have the property that repetitions in the plaintext are not detectable in the ciphertext. If two or more fragments of the input are identical, when they are combined with corresponding fragments of the pseudo-random keystreams, they yield different output fragments. This also means that an attacker cannot insert duplicate fragments of ciphertext with the hope that they are interpreted, when decrypted, as repetitions of the original content. If there is some integrity check in the contents, such repetitions would be detected as corrupted data. This property is of particular interest in audiovisual applications such as video surveillance.

Block encryption, if used in a straightforward manner, does not show this behaviour. Equal input blocks, when encrypted with the same key, produce equal output blocks. In order to conceal possible repetitions in the output bit stream, the so-called modes of operation of block ciphers are defined. The same technology is also capable of detecting forged repetitions in the

encrypted bit stream when Message Authentication Codes (MAC) are used. Examples of such modes are those known as Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB). In these modes, each block of ciphertext C_i is not obtained from the corresponding block of plaintext P_i alone, but from some combination of P_i and the previous encrypted block C_{i-1}. In this way, there is an additional input to each instance of the encryption algorithm, which is variable and propagates through the data to be encrypted, and therefore masks any possible repetition in the input blocks.

Moreover, in some modes like CFB or OFB the encryption algorithm is applied to some values computed from C_{i-1}, and the result is then combined with the plaintext block P_i. Thus, CFB and OFB effectively transform the block cipher into a stream cipher, where the block encryption algorithm is used as the pseudo-random generator. This means that in these modes the length of the input does not need to be multiple of the block length, and padding is therefore unnecessary.

If each block P_i is encrypted in combination with C_{i-1}, for encrypting the first block of plaintext P_1 it is necessary to use some C_0 block. This is called the Initialisation Vector (IV) and is simply a sequence of n bits, where n is the block length of the algorithm, which are generated randomly and used for encrypting the first block as though they were the result of a previous encryption.

A side effect of this is that using a different IV every time a new block cipher encryption is started, the output always is different even for identical inputs. In other words, duplicates are masked in the output not only within a run of the encryption system, but also from one run to another. However, this is not true in pure stream ciphers. Encrypting some input and then restarting the system and encrypting the same input again with the same key necessarily produces the same output. For this reason some stream cipher algorithms have been adapted to accept an IV thus making them parametrisable. The IV in this case determines the initial state of the keystream generator.

In either cipher type, stream or block, the IV is generally prepended or otherwise attached to the encrypted data because it is needed for correct decryption, and it is not necessary to keep it secret since an attacker does not gain any useful information by analysing the value of the IV nor comparing it with the ciphertext.

2.2. Adaptation of H.264/SVC video

H.264/AVC has a two-layered architecture [14]. The top layer, the video coding layer, derives a set of compact code sequence representing the input video sequence. The bottom layer of H.264/AVC, which is known as the Network Abstraction Layer (NAL), organises these codewords into a set of logical units called Network Abstraction Layer Units (NALUs) for optimal delivery over a given communication network. These NALUs can be parsed independently and if the reference frames are available, they can subsequently be decoded.

Evolving H.264/SVC standard [10] inherits all the features of the H.264/AVC standard. The objective of this extension is to address the increasing demand for context-aware content adaptation through simple, low complexity operations and also to improve the error resilience [9]. The standard supports a number of scalability options including spatial, temporal and Signal-to-Noise Ratio (SNR) scalabilities. The basis of the scalability in H.264/SVC is the NALU. It encodes the picture data in such a way that the adaptation can be performed by simply dropping a set of NALUs from the bit stream. Temporal scalability is achieved

through developing a hierarchical prediction structure in which the pictures belong to higher temporal resolution layers are predicted from pictures belong to the same temporal resolution layers or lower. In contrast, the spatial resolution layers are predicted from lower spatial layers and temporally neighbouring pictures. Similarly, SNR enhancement layers are predicted from the corresponding base layer and temporally neighbouring pictures. Due to the above discussed hierarchy, adaptation can be achieved through discarding NALUs defining unwanted scalability layers. The importance of these NALUs in content adaptation and specific adaptation scenarios are briefly discussed in the following subsections.

2.2.1. Types of NALUs and their importance in adaptation

Basically, there are two classes of NALUs. The first class is called the Video Coding Layer (VCL) NALUs and they carry coded representation of the base layer or an enhancement layer of a picture. Generally, an AE discards a selected set of VCL NALUs to perform the adaptation operation. However, the content of these NALUs are not altered during the adaptation operation. The second class of NALUs is known as non-VCL NALUs. These NALUs deliver supporting information, which may be required for decoding the encoded picture data or presenting them. Parameter sets (i.e., Sequence Parameter Sets (SPS), SPS extension, sub-SPS and Picture Parameter Sets (PPS)), Supplemental Enhancement Information (SEI) and Video Usability Information (VUI) are some examples of NALUs belong to this category. Amongst these NALU types, those carrying parameter sets have direct impact on the decoding process. This is because syntax elements defined in these parameter sets are needed for decoding VCL NALUs. Therefore, VCL NALUs have direct or indirect references to parameter sets. As a result these NALUs can only be discarded if none of the remaining NALUs in the bit stream refers to them. More importantly, these NALUs cannot afford to be lost because they are needed for decoding many VCL NALUs. However, SEI and VUI NALUs have no direct impact on decodability of VCL NALUs. They carry extra information to help displaying the decoded pictures. Therefore, depending on the context, this information can be dropped or altered.

2.2.2. Random packet dropping and prioritised packet dropping

Random packet dropping can be considered as adaptation to ease bottleneck situations over communication networks when routers cannot cope with high volumes of traffic that pass through them. Since each NALU can be decoded independently, the H.264/SVC bit streams offer some resilience to random packet droppings during the transmission over a lossy channel. More accurately, the loss of a VCL NALU makes only one scalability layer or a part of the layer unavailable. When the lost information is recovered with an appropriate error concealment technique, the rest of the bit stream can be decoded. However, as mentioned earlier, the loss of one or more parameter sets may prevent decoding of a large number of frames if not the entire sequence. Hence, it is necessary to make sure that these NALUs reach the destination. Various techniques such as repeated transmission of those NALUs at regular intervals, use of out-of-band secure channel, and hard coding parameters sets at the decoder have been proposed to mitigate this problem [14].

In contrast, the prioritised packet dropping makes use of the priority indices to select the packets to drop for traffic shaping. The advantage of this technique over random packet dropping is that the decoded picture quality can be improved by carefully assigning the priority indices for each NALU before transmission. H.264/SVC supports such exercise by offering a dedicated syntax element in the NALU header. Nevertheless, there is a high chance that the priority index stored in the Real-time Transport Protocol (RTP) packet header [15] is

used for this purpose since it provides more generic solution. In any case, the encoder (or any third-party entity before transmission) should assign the priority indices.

2.2.3. Systematic adaptations

Systematic adaptations are performed by specialised AEs. Even though, there is no standard to define a systematic AE, MPEG-21 Digital Item Adaptation (DIA) [16] outlines a generic architecture. This architecture, however, depends on the availability of an associated Bit-stream Syntax Description (BSD) [17] for each encoded bit stream. Nevertheless, the implementation of a systematic AE can follow any liberal architecture. An example of such a systematic AE is the JSVM bit stream extractor.

Similar to any adaptation of scalable contents, these AEs eliminate a selected set of NALUs. However, the NALU selection algorithm considers more factors than just the priority of the NALU. For example, the JSVM Bit Stream Extractor (BSE) considers the spatial, temporal and quality layer IDs available in NALU headers, and also the frame size (width and height in pixels). Since it considers the frame size, which is defined in the scalability information SEI message, the AE should be able to decode the SEI messages. A possible adaptation operation, which can be performed using the information available in NALU headers and scalability information SEI messages, is the extraction of a scalability layer with an expected spatial resolution or smaller. Moreover, these AEs can drop not only unnecessary VCL NALUs, but also the subset SPSs and PPSs which are not useful to decode remaining VCL NALUs and modify SEI messages such as scalability information and sub-sequence information. Table 1 shows the non-VCL NALUs present at the beginning of the bit stream before and after adaptation. Here, the adaptation of the Foreman test sequence (CIF, i.e., 352x288 at 30 frames per second, fps resolution) encoded with two spatial and four temporal scalability layers to extract the lowest spatial resolution (QCIF, 176x144) at highest temporal resolution (30 fps) is considered.

Table 1. Non-VCL NALUs present in the original and adapted bit streams

Original bit stream			Adapted bit stream		
NALU number	NALU type	NALU length	NALU number	NALU type	NALU length
0[†]	SEI	250	0[‡]	SEI	147
1	SPS	9	1	SPS	9
2	Subset SPS	12	2	PPS	4
3	PPS	4			
4	PPS	5			
5	PPS	5			

Since systematic adaptation techniques require some specific information from the bit stream, AEs that perform such adaptations need access to the relevant information. This information includes scalability layer IDs specified in VCL NALU headers, PPS IDs specified in the slice headers of VCL NALUs and some SEI messages.

3. Security architectures for content adaptation

None of the traditional encryption approaches consider the adaptability of scalable video. Consequently, adaptation of video encrypted with these techniques relies on trusted AEs,

[†] Scalability information SEI message
[‡] Scalability information SEI message

which decrypt the content before performing the adaptation. Novel to the technique proposed in this paper is that the encryption is transparent to scalable video adaptations. Therefore, with the introduction of the new encryption technique, there are two basic security architectures that can be considered for content adaptation: security based on trusted AEs and end-to-end security.

3.1. Trusted AE based architecture

In this architecture, the AE must perform a decryption, adaptation and re-encryption cycle in order to adapt the secured contents as illustrated in Figure 2. The cryptographic keys, which are known to the source and the legitimate end-user, must also be shared with the AE. In a possible variant scheme, the original contents can be encrypted with one key known to the source and the AE, and the adapted contents can be encrypted with another key known to the AE and the end-user.

The fact that the AE decrypts the input content prior to adaptation has the following security implications:

- Trust must be placed on the AE, because the content in unencrypted form will be available to it after decryption. Therefore, the use of the trusted AE is based on the confidence that it is not under the control of an opponent that could make illegitimate use of the adapted content.

- The trusted AE must also be robust so that the unencrypted content will never be revealed to unauthorised third parties accidentally. This robustness implies that an attacker cannot gain profit from the AE's reaction to unusual or unexpected conditions. For example, if restrictions apply to some certain content, the AE must guarantee that, if they are applicable to any form of the content, they cannot be bypassed through an adapted version, e.g. by requesting an unusual resolution.

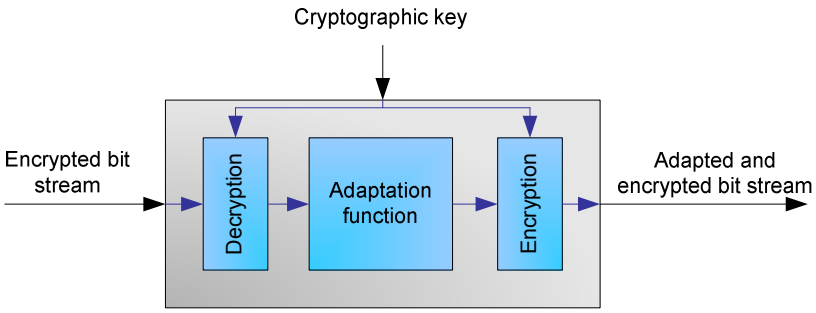


Figure 2. A trusted AE for encrypted content adaptation

When sharing the encryption keys with the AE, special care must be taken in order to prevent capture of these keys by the third parties, and to ensure that they are being sent to the authentic AE and not to a fraudulent entity impersonating it. If the keys are sent over the network, an appropriate secure key exchange protocol should be used.

3.2. End-to-end security adaptation architecture

In order to overcome the issues that may arise with trusted AEs, an alternative solution is to use adaptation techniques that do not require the AE to decrypt the input content as illustrated in Figure 3. In this way, end-to-end security is attained since no attacks to the AE will impair the protection of the content.

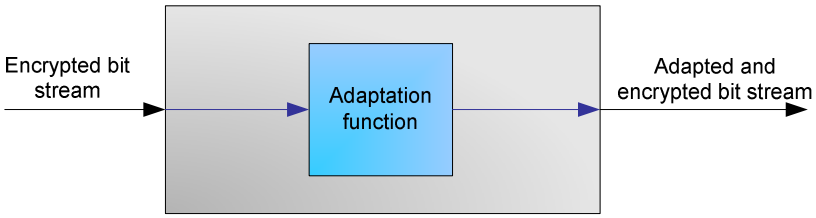


Figure 3. An AE for an end-to-end security adaptation architecture

This scenario is more secure but also more complex on the encoder side, and not as generic as the trusted AE case because not all types of adaptations can be applied "blindly" to contents, i.e., without knowing the actual value of the contents. Only those adaptations which consist of dropping parts of the content, such as spatial cropping, Region Of Interest (ROI) selection, lowering temporal resolution or discarding higher quality layers, are candidate techniques to end-to-end adaptation. The actual feasibility of these transformations depends on the encoding method used for conveying the audiovisual contents.

With regards to security, the advantages of the end-to-end architecture can be summarised as follows:

- The bit stream is not available in an unencrypted form at any moment during the adaptation, thus there is no risk that the protected content is leaked, intentionally or accidentally, out of the AE.
- Since the AE does not need to decrypt the bit stream, no cryptographic keys have to be shared or exchanged with the AE. Thus, this architecture removes the need for implementing key management protocols in the AE.
- The fact that the AE does not have access to the unencrypted content or to the encryption keys implies that no special security measures have to be applied to it. Then, there is no need to deposit trust on the AE, which in turn means that identification and authentication protocols for the AE are not necessary either.

From the performance point of view, on one hand the end-to-end architecture may require specific additional computations in order to guarantee content protection in this blind adaptation scenario (e.g., for obtaining appropriate initialisation vectors as in the technique described in Section 4 below). But on the other hand, the use of a non-trusted AE relieves this architecture of authentication and key exchange procedures with the AE, which reduces the complexity of the implementation.

4. Proposed adaptation-aware encryption method

The objective of the proposed adaptation-aware encryption algorithm is to keep the bit stream as transparent as possible for an AE in order to achieve the end-to-end protection discussed in Subsection 3.2. Therefore, the proposed technique encrypts only the carefully selected parts of the bit stream so that the syntax elements potentially carrying useful information to facilitate the adaptation operation are available unencrypted. As a result, the secured bit stream contains encrypted portions as well as unencrypted (clear) portions. Since necessary syntax elements are available clear in the bit stream, a compatible decryptor is capable of locally deriving information needed for decrypting the encrypted bit stream. In order to encrypt the selected encryptable portions of the bit stream, any standard encryption algorithm can be used. This section elaborates on the proposed method.

4.1. Security requirements for end-to-end adaptation

The proposed adaptation method is designed to meet the following security functional requirements:

- *Data confidentiality*: During the adaptation process, the audiovisual contents must be protected from disclosure to any entity, including the AE itself.
- *Concealment of data patterns*: The internal relationships between parts of the protected contents must not be observable or deducible, in particular any repetition of previously transmitted contents must not be detectable.
- *Cryptographic support*: The system must use cryptographic algorithms, which are standardised or approved by reliable organisations, with the recommended modes of operation and minimum key lengths.

This adaptation scheme does not address any key management requirements other than those applicable to the encryption of content in general. Since the AE is completely unaware of the keys used for end-to-end security, any existing key management technique can be used alongside this algorithm.

4.2. Selection of data for encrypted

In general, there are three approaches to select the portions needed for encryption. The first approach is to encrypt only a few bytes from the beginning of each NALU. Since the rest of the NALU cannot be parsed without correctly parsing the first part due to the use of variable length coding, it can be assumed that the information in the unencrypted part of the NALU is safe. Since the NALU header is not entropy coded, encrypted part should be extended at least few bytes into the rest of the NALU. However, this technique does not offer the full protection to the content since a deterministic hacker may still be able to decode the clear content using the properties of entropy coding technologies that were used to encode the data stream. Nevertheless, it must also be noted that in this case the encrypted part should be at least as long as the encryption key. Otherwise a brute-force search on the encrypted parts of the bit stream would be more effective than a brute-force search on the key. The second approach is to encrypt the entire NALU irrespective of whether the information is significant or not providing the strongest protection to the video stream.

The last approach is to encrypt a selected set of syntax elements. The encryption technique described in this paper is based on this approach. The technique should be operated carefully while selecting which syntax elements to be encrypted since the unencrypted elements may reveal enough information for an unauthorised user to guess the content of the entire video. For example, one can consider encrypting the motion vectors. However, motion compensated residual signal may carry some visual information. Therefore, such an approach may not be ideal for an application that needs absolute protection.

The proposed algorithm leaves the first part of each NALU, which spans over the NALU header and a part of the slice header, clear. The significant advantages of this approach are:

- The simplicity since complicated content analysis techniques are not necessary to select important syntax elements to be encrypted
- The greater protection against unauthorised access since all the visual information resides in the encrypted portion.

The selection of clear syntax elements is performed by considering whether any of those elements are useful for performing the adaptation operation. Adaptation scenarios discussed in Subsection 2.2 are critically evaluated to identify the required syntax elements for performing adaptation operations

For most of the adaptation operations, information in the VCL NALU headers such as the scalability layer identifications is required. Therefore, the VCL NALU headers are not encrypted. Some adaptation decisions may also benefit from certain information available from parameter sets, such as the frame size. Therefore, the parameter set identification syntax element, which can be found in the slice header of VCL NALUs, is also left clear. Need for a unique Initialisation Vector (IV) for encrypting each VCL NALU, as discussed in Subsection 4.2 below, is another reason for not encrypting some of the specific syntax elements. Furthermore, all of the syntax elements, which are available in the bit stream before the last useful syntax element, are also left clear even if they are not useful for any adaptation operation (e.g., reserved bits) in order to simplify the encryption process. Considering these factors, the syntax elements in a VCL NALU illustrated in Table 2 are identified for not encrypting. These syntax elements are available in the first part of the NALU and therefore, the rest of the NALU can simply be encrypted. Since the syntax elements shown in the table do not carry any encoded picture samples, there is no risk of exposing visual information to unauthorised users. It should be noted that some of the optional syntax elements have not been shown in this table for simplicity.

Furthermore, all of the non-VCL NALUs are also made available unencrypted in the bit stream. These NALUs are also free from encoded picture samples and therefore there is no threat of visual information leaking through these NALUs. Nevertheless, syntax elements in parameter sets are needed for parsing some of the syntax elements, such as *frame_num*, in the slice header which are useful for encryption.

4.3. Initialisation Vector (IV)

Audiovisual contents, considered as static data, can be encrypted with any type of encryption algorithm. But if these contents are to be transmitted in real time, as in a live streaming session, some algorithms are more appropriate than the others.

If a stream cipher is to be used, some synchronisation information such as a packet number needs to be sent in the clear portion of the video stream, in order to detect loss, repetition or re-ordering of packets. With this information, the receiver can detect packet loss and skip over the fragment of keystream matching the lost packets, so that decryption may continue at the right point with the next received packet. The skipped keystream may have to be generated nevertheless if a feedback generator is used in which each bit of the keystream depends on the value of the previous ones as it is common in stream ciphers.

With a block cipher, or with a stream cipher that accepts an IV, there are two main approaches: to use one single IV for the whole session, or one IV for each individual packet. Possible intermediate solutions would be based on groupings of packets and using an IV for each group. If a single IV is used, the same considerations apply as for pure stream ciphers mentioned above. Furthermore, in some modes of operation, e.g. CBC and CFB, loss of one block prevents decryption of that block and the next one encrypted with the same IV.

Table 2. Syntax elements which are made available unencrypted

	Name of the syntax element as defined in the H.264/SVC specifications	Description	Used for IV
NALU header	*forbidden_zero_bit*		
	nal_ref_idc		
	nal_unit_type	Identifies the NALU type	
	reserved_one_bit		
	idr_flag		
	priority_id	Indicates the priority of the NALU	
	no_inter_layer_pred_flag		
	dependency_id	Spatial layer identification	✓
	quality_id	Quality layer identification	✓
	temporal_id	Temporal layer identification	
	use_ref_base_pic_flag	If set, the quality layers are predicted from the base quality layer of the reference frame and otherwise higher quality layers have been used	
	discardable_flag	If set, the NALU can be discarded	
	output_flag		
	reserved_three_2bits		
Slice header	*first_mb_in_slice*	Identifies the first macroblock of the picture the slice starts from	✓
	slice_type	Slice coding type (intra, inter, bidirectional)	
	pic_parameter_set_id	Identifies the PPS corresponding to the slice	
	frame_num	An identifier for pictures[§]	✓
	field_pic_flag	If set, the slice is a slice of a coded field	
	bottom_field_flag	If set, the slice is part of a coded bottom field	✓
	idr_pic_id	When two consecutive pictures in decoding order are both Instantaneous Decoding Refresh (IDR) pictures, a different value is assigned to the latter	✓
	pic_order_cnt_lsb	An identification for the picture[**]	✓
	delta_pic_order_cnt_bottom		✓
	redundant_pic_cnt	If the slice is a redundant representation for a coded picture is a non-zero value is assigned to this syntax element.	✓

A possible additional requirement for live streaming transmissions is that participants may be able to join the session at any moment. If a block cipher with one single IV or a stream cipher is used, a new participant will need information on the updated encryption vector or the current state of the keystream generator, respectively. The latter must not be revealed to third parties or else an attacker could easily compute the rest of the keystream.

For these reasons, and because of some issues related to the use of an IV in stream ciphers [18], it is typical for encrypted streaming protocols to use block ciphers with an independent IV for each packet. This is the case in e.g. the Secure Real-time Transport Protocol (SRTP) [19], the secure version of the RTP streaming protocol (RFC 3550). In RTP, packets consist of two parts: header and payload. In SRTP a compatible header format is used, and the payload, i.e. the audiovisual content, is encrypted with a block cipher algorithm, Advanced Encryption Standard (AES) [20], using an IV constructed from certain fields of the header, one of which is a packet sequence number, thereby guaranteeing the uniqueness of the IV.

[§] This syntax element does not uniquely identify a picture in the encoded bit stream. More than one consecutive picture may share the same value for *frame_num*.

[**] This syntax element does not uniquely identify a picture in the encoded bit stream. Values may be reused at a later stage in the bit stream.

In our system, in addition to all of the previously mentioned requirements, we need to cope with content adaptation. When adaptation is performed in an end-to-end fashion, it consists basically of dropping parts of the content and perhaps duplicating certain parts (for enhanced error resilience). Therefore, we need an encryption scheme that allows decrypting the bit stream successfully even when some fragments of the encrypted content are missing, in a situation similar to that of packet losses in an unreliable network. For the same reasons explained above, we are using a block cipher algorithm with an IV derived from selected fields in the NALU header combined with a global IV, whose value is common for all of the NALUs in the same stream. This global IV is generated randomly every time a stream is to be encrypted, so that encrypting the same stream twice produces different results.

In our tests, we have used the AES algorithm, against which no realistically effective attacks are known today [21], with 128-bit keys and both in the CBC and CFB modes. In order to make sure the IV is unique for each NALU, it is constructed using a number of syntax elements from the NALU header and the slice header as shown in Table 2. In the picture level, the value of the *frame_num* syntax element may be shared among a number of consecutive pictures. When combined with the *pic_order_cnt_lsb* syntax element, which has different values for consecutive pictures, it is possible to make an identity for each picture. However, if redundant representations for a given picture are also available in the bit stream, the *redundant_pic_cnt* syntax element is used to identify each redundant NALU uniquely since the *frame_num* and *pic_order_cnt_lsb* combination remains the same for all the redundant representations. Similarly, *delta_pic_order_cnt_bottom* is necessary for identifying the top and bottom fields of an interlaced picture. In a rare case, the encoder may decide to encode a number of consecutive pictures as Instantaneous Decoding Refresh (IDR) pictures [10]. In this case, *idr_pic_id* can uniquely identify each IDR picture even if the *frame_num* and *pic_order_cnt_lsb* combination resets to zero after encoding each IDR picture.

Even if the above discussed combination of syntax elements uniquely identifies a picture in an H.264/SVC bit stream, the issue of multiple NALUs generated by encoding a picture should also be addressed in order to generate a unique IV for each NALU. It is obvious that NALUs representing different scalability layers of a given picture bear the same picture identification code generated combining the syntax elements described in the previous paragraph. Therefore, it is necessary that the *dependency_id* and *quality_id* syntax elements are also incorporated. In case of an AVC compatible base layer case, those syntax elements are both assumed to be equal to zero. Still, there is an issue: the total number of macroblocks of a given scalability layer of a picture may be distributed into more than one NALU since the length of a NALU can be in the range of just one macroblock to all of the macroblocks in the scalability layer. In this case, consecutive NALUs may have to share the same IV. Therefore, the *first_mb_in_slice* syntax element is also used to distinguish each NALU.

4.4. Encryption

Two distinctive architectures are proposed to encrypt H.264/SVC compatible bit streams. The first architecture, which is called the encoder-assisted encryption, relies on the encoder to obtain the necessary information for generating the IV and identifying the portions of the bit stream to be encrypted. Therefore, in this architecture the encryptor is tightly coupled with the encoder. The proposed architecture is shown in Figure 4. The IV generator obtains the values of syntax elements required for generating the IV for each NALU from the encoder. The encyptor passes input bits to the output clear (i.e., unencrypted) until it receives the signal from the encoder to start encrypting them. With this signalling, all of the NALU types, except for the VCL NALUs, are passed through to the output unencrypted. In case of a VCL NALU,

the first part of the NALU is passed to the output unencrypted since it carries the information needed for performing the adaptation operations and those needed for generating the IV. The rest of the NALU is encrypted. The algorithm to identify the encryptable portions of a bit stream is depicted in Figure A1 in Appendix A.

In contrast, the second architecture, which is known as the standalone encryption, is proposed for encrypting pre-encoded contents. This architecture is illustrated in Figure 5. The bit stream parser parses the first few bytes of the bit stream to extract the syntax elements to compose the IV. At the same time, it also determines the start byte of each NALU to be encrypted. Once the start location is determined, the parser signals the encryptor to encrypt the input. The same algorithm proposed for encoder-assisted encryption scenario is used for determining the encryption boundaries.

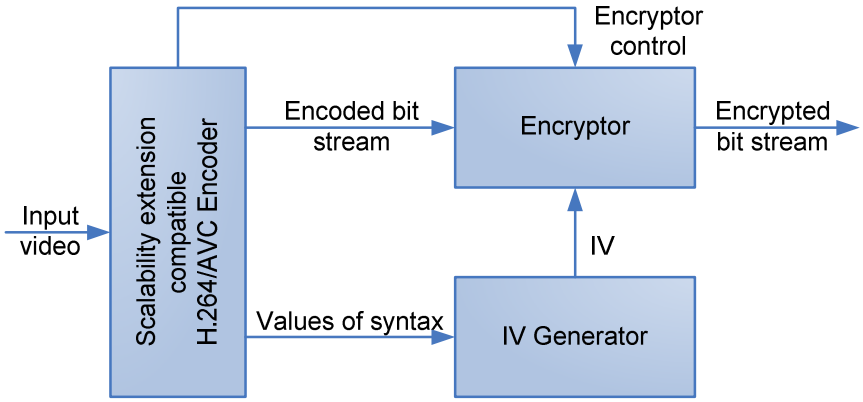


Figure 4. Encoder-assisted encryption

4.5. Decryption

Similar to encryption architectures presented in the previous subsection, two decrypting architectures are proposed. The first architecture, which is known as the decoder-assisted decrypting, depends on a H.264/SVC decoder for obtaining the required parameters for decrypting the content. Therefore, the decoder must have the understanding of which syntax elements have been left clear by the encryption technique. The proposed architecture is illustrated in Figure 6.

The decoder can parse (and decode) the NALUs, which do not carry any encoded picture data, without any extra processing since they are not encrypted. However, when a given NALU carries encoded picture data, the decoder can parse only the unencrypted syntax elements. The latter part of the NALU must be decrypted before decoding the NALU. Now the problem is how to determine the encryption boundary. Fortunately, this information can easily be obtained by parsing syntax elements known to have been unencrypted. Therefore, the decoder parses the first part of the NALU to extract unencrypted syntax elements, which are also required for generating the IV. At the same time, the decoder determines the encryption boundaries for the NALU. When the last unencrypted byte is parsed, it signals the decryptor to start decrypting the rest of the NALU. This algorithm is depicted in Figure A2 in Appendix A.

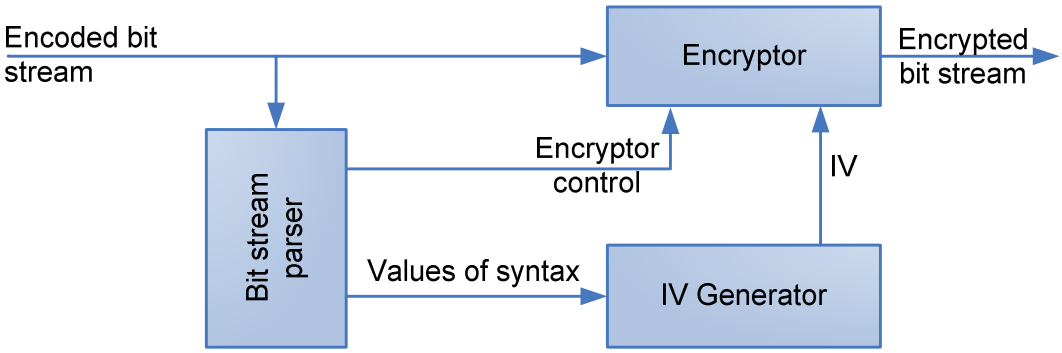


Figure 5. Standalone encryption

The drawback of the decoder-assisted decryption architecture discussed above is the need of a fully customised decoder, which is fully aware of the encrypting mechanism. This closely coupled architecture may not be practical especially when a third party decoder is used for decoding purposes. Considering this difficulty, the second decryption architecture, which is identified as the standalone decryption architecture, is proposed. This architecture is illustrated in Figure 7. In this architecture, a bit stream parser is used for extracting syntax elements for generating the IV and deriving the encryption boundary of a given NALU. Furthermore, it also identifies the clear and encrypted parts of the NALU by invoking an algorithm similar to the one proposed for the same purpose for the decoder-assisted decryption architecture. This information is passed to the decryptor through the decryptor control signal.

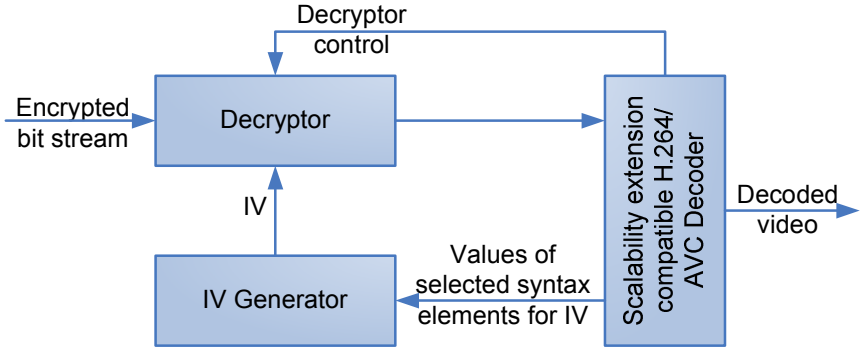


Figure 6. Decoder-assisted decryption

4.6. Security evaluation

The following properties of the proposed adaptation scheme can be considered in order to assess the fulfilment of the security requirements specified in Subsection 4.1:

- *Data confidentiality*: The AE does not need to look at the protected parts of each NALU to perform its function. Therefore if the input bit stream is encrypted the output remains encrypted, and the actual contents are never disclosed during the adaptation process.
- *Concealment of data patterns*: The use of a different encryption IV for each NALU, derived from carefully selected elements in the non-encrypted part to guarantee their uniqueness as detailed in Subsection 4.3, and the use of a random global IV for each bit stream, assure that all of the encrypted data will be uncorrelated even if the same input sequence is repeated multiple times.
- *Cryptographic support*: The proposed method makes use of symmetric block ciphers, in one of the chaining modes of operation (e.g., CBC or CFB), but does not

impose any restrictions on the actual cryptographic algorithm used or on the key length. Specific applications may choose the cryptographic engines which are best suited to their security needs.

4.7. Start code emulation prevention

The encryptor shown in Figure 4 and Figure 5 implements a standard encryption algorithm. However, this encryption process may produce specific three-byte sequences that shall not occur at any byte-aligned position in the H.264/SVC bit stream [10]. The H.264/SVC standard specifies four such codes and replacement codes for each of these codes as shown in Table 3. An AE may react to these sequences, if they are available in the encrypted bit stream, as specified by the standard resulting in undesirable behaviours. Especially, three-byte sequences 0x000000 and 0x000001 should not occur in the encrypted bit stream, since they are parts of NALU start code prefixes for applications that deliver NALU stream as an ordered stream of bytes such as MPEG-2 Systems [22]. Therefore, if any of these sequences is detected in the encrypted bit stream, the encryptor inserts an emulation prevention byte (i.e., 0x03) as shown in Table 3 to ensure that none of these forbidden three-byte sequences occurs in any NALU. This process is known as start code emulation prevention and it makes sure the encrypted bit stream is compatible for both packet oriented and byte stream oriented delivery scenarios. Analogously, the decryptor shown in Figure 6 and Figure 7 maps any occurrence of the replacement sequences into the original three-byte sequences prior to applying the decryption algorithm

Table 3. Three-byte sequences that shall not occur at any byte-aligned position in the H.264/SVC bit stream and replacement when present

Forbidden three-byte sequences	Replacement sequences when present in the bit stream
0x000000	0x00000300
0x000001	0x00000301
0x000002	0x00000302
0x000003	0x00000303

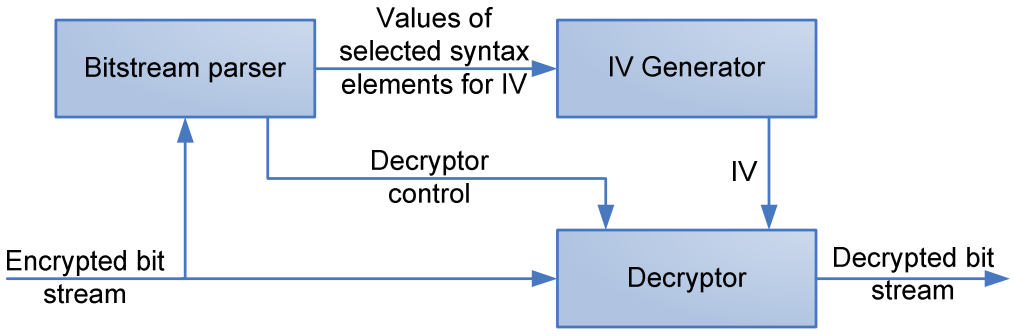


Figure 7. Standalone decryption

4.8. Signalling

The bit stream corresponding to the encrypted contents, before and after adaptation, must provide enough information for the decoder to be able to decrypt the contents. With the proposed technique for encrypting the streams, the minimum information required is the encryption algorithm and the global per-stream IV. In some cases, certain algorithm-dependent parameters may also be necessary, such as the variable key length or number of

iterations. A simple data structure, which can be delivered using any existing signalling technique used in secured content delivery, is used for including this information. Apart from these, no further parameters are required since the local IV for each NALU is algorithmically determined from its header fields.

4.9. Implications on the error resilience

One of the major advantages of H.264 standard is that it incorporates a number of error resilience features by design [23][24]. The proposed encryption technique treats individual NALUs independently that makes it possible to decrypt any NALU regardless of whether previous NALUs are available at the decryptor. Therefore, there is no known implication on any of the error resilience features available in the H.264 standard in random packet drop situations. Besides the random packet dropping, a sophisticated decoder may also be able to cope with random bit errors up to a certain extent [25]. With a stream cipher, flipping one bit in the encrypted input stream simply causes the corresponding bit in the decrypted output stream to be flipped. Therefore, these sophisticated decoding techniques can easily be used with decrypted contents.

With a block cipher in general, however, changing one bit of a block affects the whole block, so that every bit in this block will be changed with 50% probability. Therefore, error detection and correction algorithms such as [25] become increasingly ineffective. However, the use of different modes of operation can expand or reduce the error propagation. In the CBC and CFB modes, errors within a block (of length n) will produce changes in bits that can be up to $2n$ positions apart. However, the OFB mode behaves in this respect like a stream cipher, so that each single bit error produces exactly one bit change in the output.

5. Results

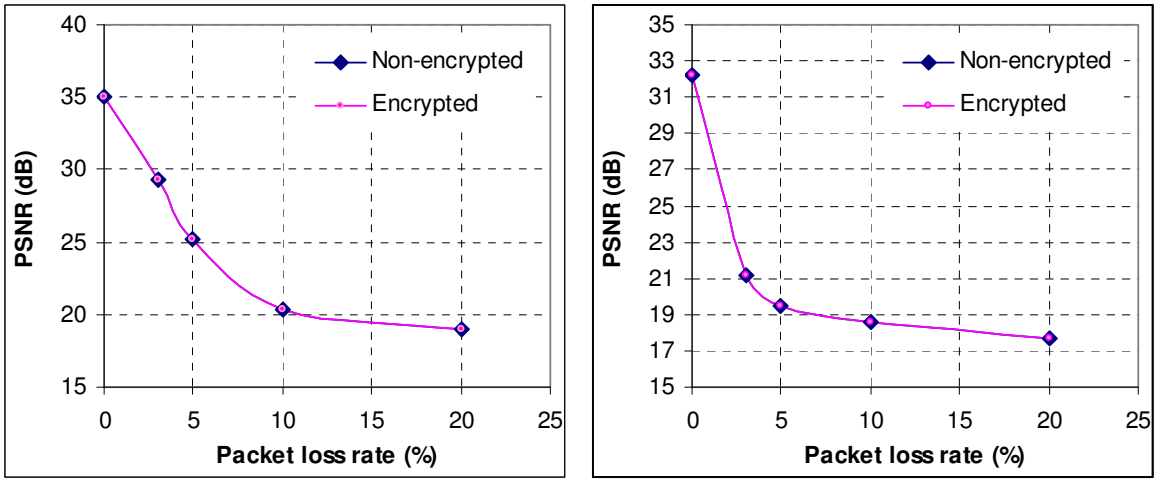
The first set of experiments was carried out aiming to investigate the transparency of encryption for AEs. Three publicly available test video sequences were encoded with the configurations depicted in Table 4. These test sequences were encoded using the H.264/SVC reference encoder (JSVM encoder). The NALU length was limited to 1000 bytes, and therefore each scalability layer produced one or more NALUs. The base layer is H.264/AVC compatible and the scalability information related to each base layer NALU was coded into a prefix NALUs [10].

Table 4. Details of the scalability structure and the length of the test sequences used for evaluating the proposed adaptation-aware encryption technique

Test sequence	Number of frames	Spatial scalability layers (width x height in pixels)	Temporal scalability layers (fps)	Total bit rate (kbps)
Forman	400	352 x 288 176 x 144	30, 15, 7.5, 3.75	255.30
Soccer	400	704 x 576 352 x 288 176 x 144	30, 15, 7.5, 3.75	453.22
CrowdRun	400	1280 x 704 640 x 352 320 x 176	60, 30, 15, 7.5	28849.14

Both the CFB and CBC encryption modes were used for encrypting the encoded content and the bit streams generated with variety of scalability structures. The JSVM BSE was used as the AE. After a number of exhaustive experiments, it was concluded that the proposed encryption technique is transparent for H.264/SVC compatible video adaptation. This can be asserted by comparing the outcome of the whole encryption-adaptation-decryption cycle with the result of conventional adaptation, i.e., without encryption. Since both results are the same, it can be concluded that the goal of transparency has been satisfactorily achieved.

Subsequently, a similar set of experiments were carried out for validating the algorithm for random and prioritised packet dropping scenarios. The criteria for validation were, as in the case of transparency, comparison of results between the setup with encryption and that without encryption. In order to demonstrate the effect of random packet losses to encrypted bit streams, a simulation study was carried out using an IP channel model. The IP channel model was implemented using the AVC/SVC loss simulator described in [26] and ITU-VCEG loss patterns [27].The test conditions specified in [28] are observed during the simulation study. Moreover, the error concealment algorithm used in this experiment considers all the lost macroblocks coded with the BLSkip mode [14]. The decoded quality of the encrypted and non-encrypted bit streams, which are received over the lossy channel, is compared in Figure 8. These bit streams are encoded with IPPP temporal prediction structure and have four temporal scalability levels. Experimental results clearly verify that random losses have little or no impact on the decryptability of the received bit streams.



(a) *(b)*

Figure 8. Effect of transmitting the encrypted bit stream over a lossy IP channel for (a) Foreman and (b) Crowdrun test sequences

Theoretically, the proposed encryption technology must also be resilient to any prioritised packet dropping scenario. To test this condition, an experiment was carried out in this kind of a scenario, in which the Foreman test sequence was encoded with two levels of temporal and four levels of quality scalability. Each scalability layer is assigned the priority according to the importance so that dropping packets from the lower priority levels minimally affects the quality of the decoded video. Hence, the base layer of the bit stream is assigned the highest priority (i.e., priority = 7). The quality enhancement layers of the lower temporal scalability layer (i.e., 12.5 frames per second, fps) are assigned with the next three priority levels. Moreover, the base layer of the highest temporal layer is assigned priority = 3 and the quality

enhancement layers are assigned the next three priority levels. In our test, it is assumed that the router drops the lower priority packets to recover from congestion. If the congestion is light, only the lowest priority level is dropped. Table 5 compares the resulting bit rates and objective qualities when each quality layer is dropped from the non-encrypted and encrypted bit streams. These results show that the encryption algorithm performs well under the prioritised packet dropping scenario.

Table 5. Evaluation of the effect of encryption on the prioritised packed dropping scenario

Dropped priority level	frame rate (fps)	Non-encrypted		Encrypted	
		Bit rate (kbps)	PSNR (dB)	Bit rate (kbps)	PSNR (dB)
none	25	1,842	37.14	1,842	37.14
7	25	1,554	36.20	1,554	36.20
6	25	1,250	35.34	1,250	35.34
5	25	741	33.92	741	33.92
4	12.5	723	37.69	723	37.69
3	12.5	602	36.63	602	36.63
2	12.5	480	35.72	480	35.72
1	12.5	298	34.44	298	34.44

The objective of the next set of experiments is to assess the processing and bit rate overheads due to the proposed encryption technology. Same test setup used for assessing the transparency for AEs was also used for these experiments. Table 6 shows the CFB encryption performance for the selected bit streams. Here, AES-128 algorithm is used for encrypting the data streams. This experiment was carried out on a 3 GHz Pentium 4 dual-core machine running a Linux operating system. It shows that over 95% of the total data bytes have been encrypted. It also shows that when the bit rate is smaller, the percentage of encrypted bytes reduces. This is because the length of NALU header and the slice header become increasingly dominant at lower bit rates. Moreover, encryption time shown in Table 6 indicates that the processing overhead per frame is negligible.

Table 6. Encryption performance

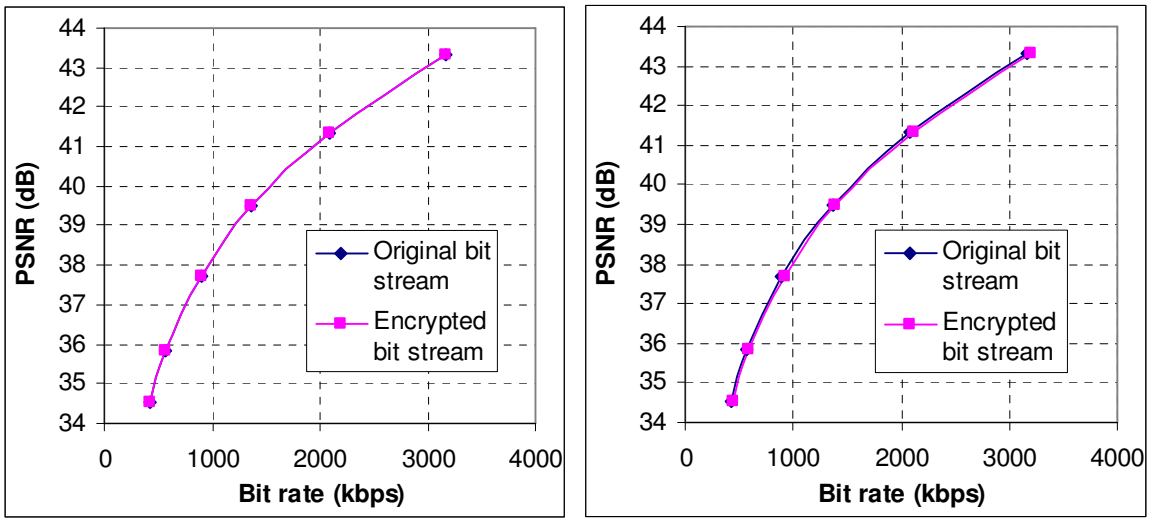
		Foreman	Soccer	CrowdRun
Total bytes		425500	755367	24040953
Bytes encrypted	Number	411493	738601	23686537
	Percentage	96.70%	97.80%	98.50%
Bytes not encrypted	Number	14007	16766	354416
	Percentage	3.30%	2.20%	1.50%
Encryption overhead	Total time (ms)	17	26	665
	Bytes/s	24.2×10^6	28.4×10^6	35.6×10^6
	Per frame processing time (μs)	77.5	117.5	1797.5

Even though the CFB encryption mode does not have a bit rate penalty, the CBC encryption has a bit rate overhead as shown in Table 7, which is a direct consequence of the padding algorithm that is applied in CBC for making the input data length a multiple of the cipher block length. It should be noted that expected theoretical average padding in an algorithm of block length N is $(N + 1)/2$. Since $N = 16$ bytes in our experiments, the average padding length should be 8.5 bytes. The results shown in Table 7 clearly agree with this theoretical

average. Furthermore, the Rate-Distortion (RD) due to each encryption mode is illustrated in Figure 9. According to Figure 9 (a), the CFB mode does not have any RD penalty. In contrast, when the CBC mode encryption is used, there is an RD penalty as shown in Figure 9 (b). Again, this is caused by the padding inserted in CBC mode. However, this RD penalty is negligible.

Table 7. CBC encryption overheads introduced to the bit stream

	Foreman	Soccer	CrowdRun
Input length	425500	755367	24040953
Encrypted length	433992	765470	24261093
Total extra bytes	8492	10103	220140
Encrypted NALUs	984	1186	25419
Avg. extra bytes / NALU	8.63	8.52	8.66



(a) *(b)*

Figure 9. RD performance of the Foreman test sequence before and after (a) CFB mode and (b) CBC mode encryption

6. Conclusions

This paper has presented a proposed adaptation-aware encryption concept and discussed the enabling technologies for encrypting H.264/SVC compatible video. The proposed technique enables end-to-end transparency for scalable video adaptation. Therefore, it is possible to reap the advantages of scalability in video coding without compromising the content security since the AE does not need to decrypt the content. This objective was achieved by leaving syntax elements required for performing the adaptation operation clear (i.e., unencrypted). Moreover, some of the clear syntax elements are also used for generating the IV for the encryption process. The transparency of the encrypted bit streams was successfully validated for systematic scalable video adaptations as well as random and prioritised packet dropping scenarios through a comprehensive set of experiments. Experimental results have shown that the proposed technique incurs negligible processing overhead.

A disadvantage of the proposed encryption techniques is its strong dependency on the H.264/SVC standard. Further experiments are being carried out to exploit MPEG-21 Bit-stream BSD to develop video-coding-agnostic encryption technologies.

Acknowledgments
The work presented has been developed within VISNET II, a European Network of Excellence (http://www.visnet-noe.org), funded under the European Commission IST FP6 programme. Authors would also extend their sincere gratitude to Ubong Ukommi of I-Lab, CCSR, University of Surrey, UK for his invaluable help for performing validation experiments.

Reference:

[1] F. Pereira, "A triple user characterization model for video adaptation and quality of experience evaluation", in *Proc. 7th IEEE Multimedia Sig. Process. Workshop*, Oct. 2005, pp. 1-4.
[2] P. Reichl, "From 'quality-of-service' and 'quality-of-design' to 'quality-of-experience': A holistic view on future interactive telecommunication services", in *Proc. 15th Int. Conf. Software, Telecommun. and Computer Networks*, Sep. 2007, pp. 1-6.
[3] F. Pereira and I. Burnett, "Universal Multimedia Experiences for Tomorrow", *IEEE Sig. Process. Mag.*, vol. 20, no. 2, Mar. 2003, pp. 63-73.
[4] I. Burnett, R Van de Walle, K. Hill, Bormans, J. F. Pereira, "MPEG-21: Goals and Achievements," *IEEE Multimedia*, vol. 10, no. 6, pp. 60-70, Oct.–Dec. 2003.
[5] R. Iqbal, S. Shirmohammadi, A. El Saddik, "Secured MPEG-21 Digital Item Adaptation for H.264 Video," in *Proc. IEEE International Conference on Multimedia and Expo*, July 2006.
[6] R. Iqbal, S. Shirmohammadi, A. El Saddik, "Compressed-Domain Encryption of Adapted H.264 Video," in P*roc. of Eighth IEEE International Symposium on Multimedia*, San Diego, CA, pp. 979-984, Dec. 2006.
[7] R. Iqbal, S. Shirmohammadi, A. El Saddik, ''A Framework for MPEG-21 DIA Based Adaptation and Perceptual Encryption of H.264 Video,'' in *Proc. SPIE/ACM Multimedia Computing and Networking Conf.*, vol. 6504, article 650403, SPIE, 2007.
[8] R. Iqbal, S. Shirmohammadi, A. El Saddik, and J. Zhao, "Compressed-domain video processing for adaptation, encryption, and authentication," *IEEE Multimedia*, vol. 15, no. 2, pp. 38-50, April-June 2008.
[9] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, Sep. 2007, pp. 1103-1120.
[10] *Advanced video coding for generic audiovisual services*, Draft revised ITU-T Recommend. H.264, Jun. 2008.
[11] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: Handbook of Applied Cryptography. CRC Press, 1996. (Available online at http://www.cacr.math.uwaterloo.ca/hac/)
[12] B. Schneier, *Applied Cryptography*, 2nd ed. John Wiley & Sons, 1996.
[13] N. Ferguson and B. Schneier, *Practical Cryptography*, John Wiley & Sons, 2003.
[14] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Recommend. H.264, Mar. 2005.
[15] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, RTP Payload Format for H.264 Video, RFC 3984, Feb. 2005.
[16] "Information Technology – Multimedia Framework (MPEG-21) – Part 7: Digital Item Adaptation," ISO/IEC Standard, ISO/IEC 21000-7:2007, November 2007.
[17] G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers and M. Amielh, "Bitstream Syntax Description: A Tool for Multimedia Resource Adaptation

within MPEG-21," *EURASIP Signal Processing: Image Communication J.*, vol. 18, no. 8, pp. 721-747, 2003.
[18] Erik Zenner, "Why IV Setup for Stream Ciphers is Difficult", in *Proc. Dagstuhl Seminar "Symmetric Cryptography",* 2007. (Available online at http://www.erikzenner.name/docs/2007_Dagstuhl_Paper.pdf)
[19] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman. RFC 3711: The Secure Real-time Transport Protocol (SRTP). IETF, 2004. (Available online at http://www.ietf.org/rfc/rfc3711.txt)
[20] FIPS Publication 197, Specification for the Advanced Encryption Standard (AES). FIPS, 2001. (Available online at http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf)
[21] "National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information," National Security Agency, CNSS Policy No. 15, Fact Sheet No. 1, June 2003. (Available online at http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf)
[22] ISO/IEC, "Generic Coding of Moving pictures and Associated Audio: Systems," (MPEG-2 Systems Specification), ISO/IEC 13818-1, Nov. 1994.
[23] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645- 656, July 2003.
[24] T. Stockhammer, M. M. Hannuksela, and S. Wenger, "H.264/AVC in wireless environments," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 657- 673, July 2003.
[25] E. Khan, S. Lehmann, H. Gunji, and M. Ghanbari, "Iterative error detection and correction of H.263 coded video for wireless networks," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 14, no. 12, pp. 1294-1307, 2004.
[26] Y. Guo, H. Li, and Y. Wang, "SVC/AVC loss simulator donation", JVT input document, JVT-Q069, Oct. 2005.
[27] S. Wenger, "Error patterns for internet experiments," ITU VCEG Q15-I-16r1, 2002.
[28] Y.-K. Wang, S. Wenger, and M. M. Hannuksela, "Common conditions for SVC error resilience testing," JVT Output Document, JVT-P206, July 2005.

Appendix A: Flow diagrams

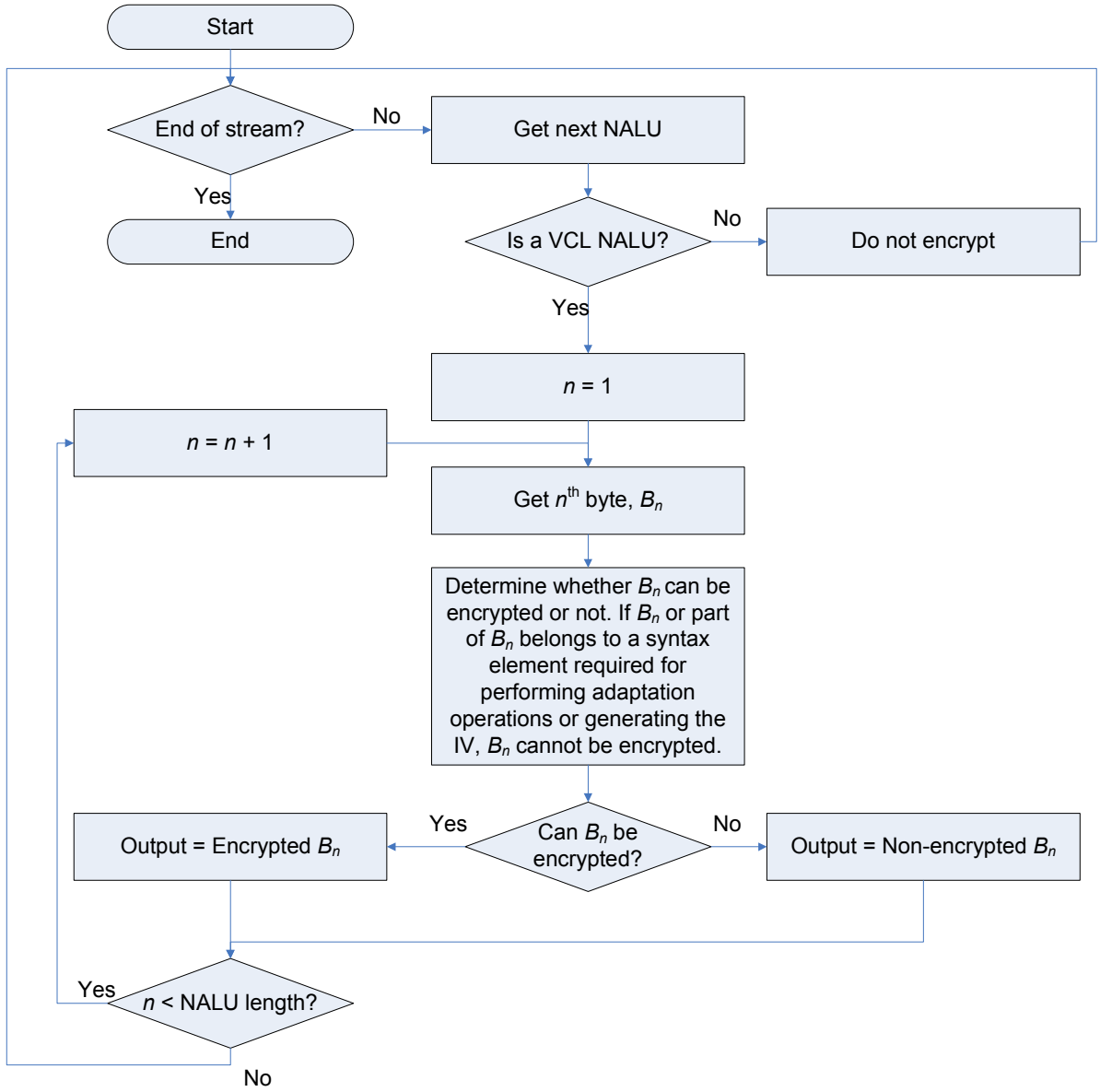


Figure A1. The algorithm to determine the encryption boundaries

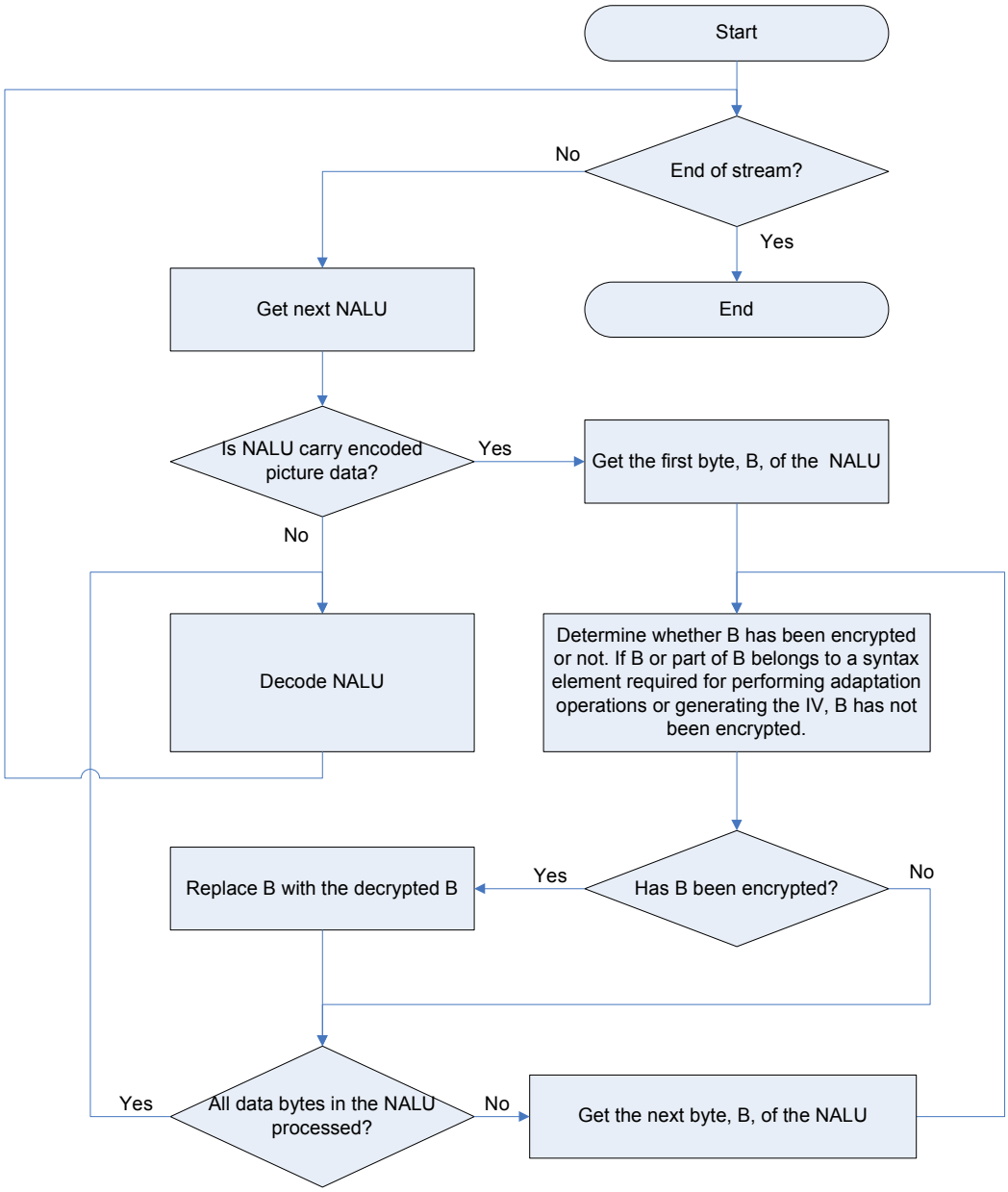


Figure A2. The decryption algorithm for decoder-assisted decrypting a bit stream