Secure Delivery of Compressed Audio by Compatible Bitstream Scrambling

Eric Allamanche
Jürgen Herre

Fraunhofer Institut für Integrierte Schaltungen
Am Weichselgarten 3
91058 Erlangen, Germany

Abstract

The recent technological revolutions in both networking technology (the Internet) and highly efficient perceptual audio coding algorithms (e.g. MPEG audio) have stimulated attractive new applications but also a considerable amount of music piracy. In order to enable a controlled distribution of multimedia content, classical data encryption techniques have been proposed to provide a "secure envelope" around the multimedia payload by means of ciphering all or parts of it. This paper presents a novel approach of securing compressed digital audio by integrating both ciphering and source coding methods into a single scheme.

1 Introduction

The recent technological revolutions in both networking technology (the Internet) and highly efficient perceptual audio coding algorithms (e.g. MPEG audio [1, 2, 3, 4]) have stimulated attractive new applications but also a considerable amount of music piracy. In order to enable a controlled distribution of multimedia content [5, 6], classical data encryption techniques have been proposed to provide a "secure envelope" around the multimedia payload by means of enciphering all or parts of it. In this way, access to the payload (i.e. decoding of the bitstream) is possible only for authorized persons who are in the possession of the proper key for deciphering. Since such protection

schemes can be applied to any kind of data and particularly to compressed digital audio, deciphering and decoding are two independent processes. A direct consequence of such a protection strategy is that no guarantee can be given about the syntactical correctness of the enciphered bitstream. In contrast, the approach presented in this paper only enciphers well defined parts of the audio bitstream while leaving its syntax intact.

The paper will first review the classical approach of data enciphering and then introduce the basic concepts of compatible bitstream scrambling. In particular, Section 3 will discuss the underlying encoding, decoding and transcoding processes as they can be applied to state-of- the-art audio coding schemes. In Section 4, some properties of this protection paradigm will be briefly discussed. Section 5 will deal with the practical implementation of compatible bitstream scrambling in the MPEG-2 Advanced Audio Coding (AAC) coder. A conclusion will finally summarize the main ideas presented in this paper.

2 Enciphering Audio Bitstreams

This section discusses the application of classical enciphering techniques in conjunction with audio coding as well as the principles of compatible bitstream scrambling.

2.1 The secure envelope approach

The commonly used approach of protecting data is based on cryptographic techniques, like those described in [7]. The basic idea is to "package" the content to be protected (payload) into a secure container by enciphering parts or all of the payload. This kind of protection is therefore referred to as the *secure envelope approach.*

Since such ciphers are able to secure any kind of data (text, images, files,...), they are also suited for protecting compressed audio bitstreams. A bitstream can be interpreted as a concatenation of different bit fields, each one having a particular semantic meaning and thus a well-defined effect during the decoding process. Since a cipher is not aware of these bit field definitions, the resulting protected bitstream will generally not constitute a valid bit stream according to the defined syntax.

The application of a secure envelope cipher for encoding and decoding audio signals is depicted on Figures 1 and 2. In contrast to other applications requiring full enciphering of the payload, such as *e-commerce*, audio bitstreams only need to be partially protected to prevent illicit listening.

Thus, the additional processing power can be kept within acceptable limits if only selected parts are enciphered.

These protection schemes have shown their merit and are nowadays widely used in securing information of all kind.

2.2 The soft envelope approach

Rather than trying to encipher the raw data, like in the secure envelope technique, the presented approach aims at protecting the information which is carried through the bitstream. The consequence is that the bitstream syntax is not affected by the protection and thus remains valid (i.e. decodable) according to the bitstream definition of the audio codec used. This property guarantees a seamless playback of the compressed audio material. Through this tight integration, the cryptographic process and the audio coding are two inseparable units. This property is illustrated on Figure 3 for the soft envelope audio encoder and on Figure 4 for the soft envelope audio decoder.

Provided the right key and the appropriate source decoder, the decoded bitstream will result in a full quality audio output. This bitstream can even be decoded properly if the decoder is not equipped with the appropriate deciphering circuitry. In this case, (or if the user's key does not match the one of the bitstream) the resulting audio signal will represent the audio content at a deteriorated quality level thus enabling a pre-listening of distributed material. This soft envelope enciphering approach will be referred to as *compatible bitstream scrambling* throughout the rest of this paper.

Like in the case of the secure envelope approach, this technique also requires a pair of keys. However, the issues involved in the management of these keys will not be addressed in this paper. The reader should consult [7] for further informations concerning this topic.

Both approaches, the traditional secure envelope and the compatible bitstream scrambling technique presented in this paper, may be used as security technologies in applications according to the MPEG-4 *Intellectual Property Management and Protection* (IPMP) concept [8, 9].

3 Principles of Compatible Bitstream Scrambling

The basic ideas of the compatible bitstream scrambling applied to audio signals are the integration of the enciphering/deciphering process within the encoder/decoder. It is therefore useful to review the main building blocks

of state-of-the-art audio coding schemes. Figure 5 illustrates the generic structure of a perceptual audio coder.

The main building blocks are:

- An *analysis filterbank* maps the input time signal into subsampled spectral components. Examples of commonly used filter banks are polyphase filter banks [10] and filter banks based on the modified discrete cosine transform (MDCT) [11].
- A *perceptual model* calculates an estimate of the signal's time and frequency dependent masking threshold. This threshold depends on the spectral and temporal structure of the signal [12].
- A *quantization* unit quantizes the spectral coefficients. The quantization error should be shaped to be below the estimated masking threshold in order to be inaudible.
- A *noiseless coding* stage performs an entropy coding of the quantized spectral coefficients by taking advantage of statistical properties.
- A *bitstream multiplexer* assembles the quantized and coded spectral coefficients as well as additional side information into one bitstream according to the defined bitstream syntax.

The decoding process is performed as the inverse of the encoding operation. The basic structure of a perceptual audio decoder is illustrated in Figure 6. The blocks are the counterpart of those in the encoder, except for the perceptual model. The decoding process is straightforward and requires less processing power than the encoder, since no perceptual model has to be estimated.

3.1 Compatible bitstream scrambling encoder

The compatible bitstream scrambling is carried out during the encoding process. As can be seen from Figure 7, the enciphering unit is embedded in the noiseless coding kernel. At this point, manipulations can be performed directly on the quantized spectral coefficients, on the codewords resulting from the entropy coding or on both. Section 5 will give more detailed insights into possible manipulations. This embedding guarantees that the resulting bitstream is fully compliant in its syntax.

It is evident that the manipulations (i.e. the scrambling operations) realized during the encoding process must be exactly reversible if no further

distortions (in addition to those introduced by the perceptual coding process) are tolerated. To this end, a pseudo random number generator is employed during both scrambling and descrambling to control the sequence of manipulations executed. More specifically, each bit sequence generated by this generator is mapped to a particular manipulation, such that it can be undone during the descrambling phase in the decoder. To enable correct descrambling only for holders of the secret key, the operation of the pseudo random number generator is defined depending on the value of the key (e.g. the key might be used as a seed value for the generator).

3.2 Compatible bitstream scrambling decoder

The structure of a compatible bitstream scrambling decoder is depicted in Figure 8. All inverse operations have to be carried out in reverse order to restore the audio data. Accordingly, the deciphering stage is located in the noiseless decoding unit.

Provided that both keys (enciphering and deciphering) are compatible, the deciphering will restore the original audio data. Otherwise, the bitstream will still be parsed and decoded but the reproduced sound will contain audible distortions.

3.3 Compatible bitstream scrambling transcoder

In some applications, transcoding of a bitstream enciphered with *key 1* into another bitstream enciphered with *key 2* is of interest. This is, for example, the case in client-server systems where the server, which holds the bitstream data base, has to deliver a protected bitstream with a personalized key to a client. The on-the-fly conversion of the bitstream from one key to another (i.e. deciphering with *key 1* and subsequent enciphering with *key 2*) can be done with the compatible bitstream scrambling transcoder. The structure of such a transcoder is depicted in Figure 9. It is evident that the transcoding process only requires a fraction of the complexity of an encoder or decoder because of the absence of the analysis/synthesis filterbanks and the quantization/inverse quantization units. The simplicity of this transcoding process makes the compatible bitstream scrambling concept well-suited for transaction tagging.

This scheme can also be applied to secure already existing but unprotected bitstreams. In this case, the deciphering with *key 1* is omitted since the plain text bitstream is already available.

4 Properties of Compatible Bitstream Scrambling

This section summarizes a number of properties inherent to the compatible bitstream scrambling approach and compares them to the traditional secure envelope techniques.

- Seamless playback: Due to the intact syntax of the scrambled bitstreams, seamless playback is guaranteed in all cases, both with and without the correct key. In contrast, playing back conventionally enciphered bitstreams without proper deciphering will produce unpredictable results and may even cause damage to the loudspeakers or the listener's ears.

- Pre-listening capability: Due to the seamless playback property and the adjustable amount of distortion caused by the enciphering, compatible bitstream scrambling allows a pre-listening of the protected material, albeit with a degraded playback quality.

- Transaction tagging: A compatible bitstream scrambling transcoder enables on-the-fly transaction tagging from one key to another. This operation can be easily performed in real time due to its low complexity.

- Plain text bitstream unavailable: Since the enciphering/deciphering process cannot be separated from the encoding/decoding operation, the plain text bitstream is available at no point in the processing chain. In contrast, within a system using the traditional secure envelope approach deciphering of the payload and subsequent audio decoding are carried out as two separate stages in the decoding process. Thus, the plain (unciphered) bitstream is available after the first stage which represents a weak point within a secure architecture.

- Complexity of attack: Since the plain text bitstream is unavailable, attacking the scrambled bitstream parts requires profound knowledge of both the coding scheme and its bitstream syntax as well as the scrambling technique used.

- Audio coders: Compatible bitstream scrambling can be easily applied to almost every perceptual coding scheme because of its underlying technical principles (e.g. MPEG-1/2 Layers 1-3 [1, 2], MPEG-2 AAC [3], MPEG-4 [4], Dolby AC-3 [13]).

- Open architecture: Since compatible bitstream scrambling is based on secret key methods (in contrast to so-called "security by obscurity" cryptography), it is well-suited as part of open architectures.

5 Compatible Bitstream Scrambling of MPEG-2 AAC Bitstreams

This section provides a more detailed description of exemplary operations which can be carried out within the MPEG-2 AAC audio coding scheme [3, 14] in order to implement the compatible bitstream scrambling technique. To this end it is necessary to take into account the specific structure of the noiseless coding kernel employed in MPEG-2 AAC. While a general overview of the AAC coder is given in [14], its Huffman code based noiseless coding kernel is described in more detail in [15].

The main selection criteria for the scrambling manipulations are to provide uncorrupted bitstream syntax and to allow for the control of the amount of degradation that results from decoding without the correct key. As will be illustrated below, the first goal can be achieved by careful manipulation of the quantized and coded data, while the latter one can be obtained by limiting the range of the manipulations.

Assuming that the most preferable type of distortion would be a smooth distortion without disturbing "blops" or similar artifacts, it seems reasonable not to alter side information like scalefactors which could result in major changes in the spectral envelope and overall energy of the signal. Thus, the focus of the following investigations is on manipulations carried out on the quantized spectral coefficients produced by the AAC coder. More specifically, the manipulations may include one or more of the operations discussed subsequently.

5.1 Bitwise XOR with spectral coefficients

A very simple way of manipulating quantized spectral coefficients can be achieved by means of a bitwise *exclusive or* (XOR) operation with some bit mask which might e.g. be derived from the output of the pseudo random number generator. In order not to grossly distort the statistics of the signal, this bit mask is applied to the least significant bits (LSBs) of the quantized spectral coefficients. This will result in a new coefficient with some degree of distortion. To restore the original quantized coefficient, it suffices to perform a bitwise XOR with the same bit pattern as during the encoding. These operations are illustrated in Figure 10. Subjective informal testing of the

perceptual degradations showed that the distortions produced for incorrect descrambling depend on the type of audio material encoded and may, for some cases, not be strong enough to discourage illegal listening.

Since the bitwise XOR operation is carried out on the quantized coefficient prior to the noiseless coding, the number of bits needed to encode these spectral coefficients may vary depending on the bit patterns.

5.2 Sign scrambling

An alternative method is the manipulation of the sign information of the real-valued quantized spectral coefficients. Again, an XOR operation with a (key-dependent) bit pattern can be used to both scramble and descramble the spectral content such that e.g. the sign of a spectral coefficient is toggled if a "1" is encountered in the corresponding bit of the bit pattern.

Within the AAC noiseless coding kernel these quantized values may either be coded by *signed Huffman codebooks* or by *unsigned Huffman codebooks*, i.e. a Huffman codeword may either be assigned to pairs or quadruples of quantized values or else represent their magnitudes only. In the latter case, the sign information is appended to the Huffman codeword as a number of "sign bits" which allows for arbitrary changes of the sign information without a change in the number of bits needed to represent the scrambled signal.

The perceived degradations caused by sign scrambling are in general more annoying than the degradations due to simple XOR of the LSBs, but may still vary considerably with the type of audio material coded.

5.3 Permutations of spectral coefficients and codewords

Instead of manipulating the magnitudes and the signs of the spectral coefficients as discussed so far, it is also possible to manipulate the position (i.e the corresponding frequency) of these coefficients. This can be achieved by permuting all or groups of spectral coefficients, as illustrated in Figure 11. Again, the permutation rule must be identical during the enciphering and the deciphering processes and permutation operations are carried out depending on the output pattern of a (key-dependent) pseudo random number generator.

In the case of MPEG-2 AAC, the use of 2- or 4-dimensional Huffman codebooks also dictates that tuples (pairs or quadruples) of spectral coefficients corresponding to one Huffman codeword are treated as "atoms" and shuffled together in their position. Equivalently, the operation can be implemented by reordering of the Huffman codewords after the noiseless coding stage. Generally, arbitrary permutations of atoms can be used within each

group of adjacent spectral coefficients which is coded with a common Huffman code table. In this case, the number of bits occupied by the scrambled bitstream remains constant.

In comparison with other previously discussed options for scrambling, reordering of spectral coefficients is probably the most interesting type of manipulation because it enables both an unchanged bitrate and a good control of the severity of the scrambling distortions which would be perceived by an unauthorized listener. If no further limitations are imposed on the range of permutation (i.e. the maximum distance between the new and the original location of spectral coefficients) rather large impairments are observed, independently of the type of audio material. With increasing restriction on the permutation range, the audible scrambling distortion tends to assume a smooth but nonetheless still annoying character.

5.4 Further considerations

As was mentioned previously, the discussed manipulations can be applied individually or in combination with each other. This may result in a greater protection at the expense of more audible distortions and increased computational complexity. Furthermore, the frame length invariance property may not be preserved, which in some situations may not be desirable.

The above overview of possible manipulations in the field of compatible bitstream scrambling is certainly not exhaustive. One should keep in mind that the protection achieved always corresponds to complexity and processing power, which must be kept as low as possible, if this scheme is to be employed in low-power applications, such as portable devices. These possibilities also permit to vary the trade-off between complexity and security.

From the above examples for manipulations it is evident that similar operations can be carried out for other coding schemes, both with or without entropy coding kernel. In the latter case, the restrictions which are imposed by the constant bitrate constraint do not apply and thus further simplify the design of the scrambling system.

6 Conclusions

This paper presented a novel technique for securing compressed digital audio, called compatible bitstream scrambling. The approach integrates both enciphering and source coding into a common framework and in this way avoids producing a plain text bitstream at some point in the system which is a weakness in traditional secure envelope schemes.

Protection is achieved by means of compatible manipulations of the bitstream syntax elements. The concept allows for playback of the protected content without the correct key (pre-listening). The amount of subjective degradation in audio quality can be controlled within a wide range, according to the content author's preference. Due to its low computational requirements, compatible bitstream scrambling is well-suited for portable devices and other low cost real-time appliances. The transcoding capability enables on-the-fly tagging and thus personalization of audio bitstreams.

Finally, the implementation of compatible bitstream scrambling in the MPEG-2 AAC coder was illustrated by discussing a few of the possible manipulations that can be carried out within the AAC noiseless coding kernel in order to implement compatible bitstream scrambling.

Although the presented manipulations were discussed in the context of MPEG-2 AAC, they are not specific to this coder, but can easily be applied in a similar fashion to almost any other state-of-the-art audio coding schemes, including MPEG-1, MPEG-2 and MPEG-4.

References

[1] ISO/IEC JTC1/SC29/WG11 Moving Pictures Expert Group. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s: Audio. International Standard 11172-3, 1992.

[2] ISO/IEC JTC1/SC29/WG11 Moving Pictures Expert Group. Generic coding of moving pictures and associated audio: Audio. International Standard 13818-3, 1994.

[3] ISO/IEC JTC1/SC29/WG11 Moving Pictures Expert Group. Generic coding of moving pictures and associated audio: Advanced audio coding. International Standard 13818-7, 1997.

[4] ISO/IEC JTC1/SC29/WG11 Moving Pictures Expert Group. Coding of audio-visual objects: Audio. International Standard 14496-3, 1999.

[5] Secure Digital Music Initiative (SDMI). The SDMI Homepage, `http://www.sdmi.org`.

[6] Open Platform Initiative for Multimedia Access (OPIMA). The OPIMA Homepage, `http://drogo.cselt.stet.it/ufv/leonardo/opima/`.

[7] B. Schneier. *Applied Cryptography.* John Wiley & Sons Inc., second edition, 1998.

[8] ISO/IEC JTC1/SC29/WG11 Moving Pictures Expert Group. Coding of audio-visual objects: Systems. International Standard 14496-1, 1999.

[9] J. Lacy, N. Rump, T. Shamoon, and P. Kudumakis. MPEG-4 Intellectual Property Management & Protection (IPMP) Overview & Applications. In *Proceedings of the AES 17th International Conference on High Quality Audio Coding*, Florence, Italy, September 1999.

[10] K. Brandenburg and G. Stoll. The ISO/MPEG-audio codec: A generic standard for coding high quality digital audio. In *92th AES Convention*, Vienna, 1992. Preprint 3336.

[11] J. Princen, A. Johnson, and A. Bradley. Subband / Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation. In *IEEE ICASSP*, pages 2161–2164, 1987.

[12] E. Zwicker and H Fastl. *Psychoacoustics - Facts and Models.* Springer Verlag, Berlin, 1990.

[13] M. Davis. The AC-3 Multichannel Coder. In *95th AES Convention*, New York, 1993. Preprint 3774.

[14] M. Bosi, K. Brandenburg, S. Quackenbush, K. Akagiri, H. Fuchs, J. Herre, L. Fielder, M. Dietz, Y. Oikawa, and G. Davidson. ISO/IEC MPEG-2 Advanced Audio Coding. *Journal of the AES*, 45(10):789–814, October 1997.

[15] S. Quackenbush. Noiseless Coding of Quantized Spectral Components in MPEG-2 Advanced Audio Coding. In *IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, 1997.

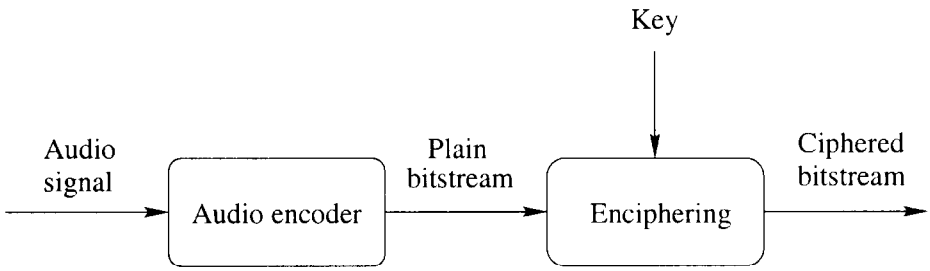


Figure 1: Structure of a secure envelope audio encoder

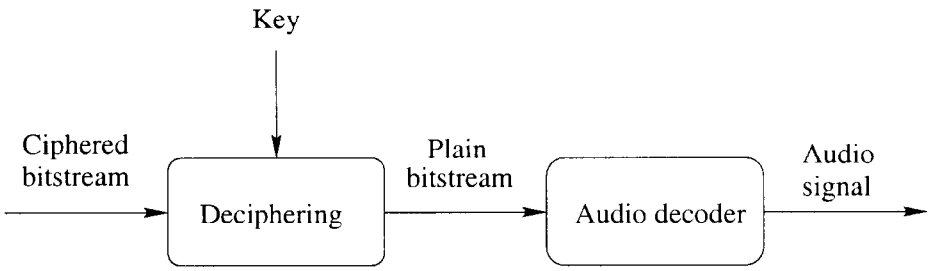


Figure 2: Structure of a secure envelope audio decoder

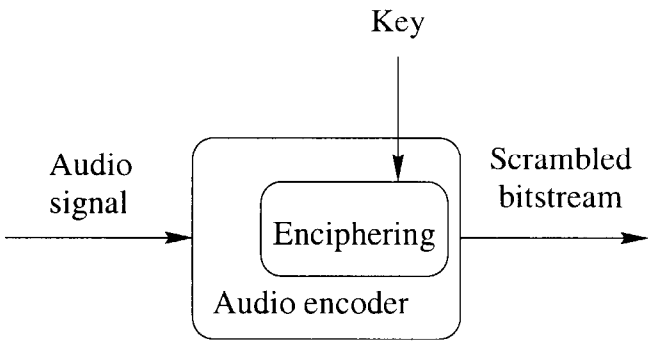


Figure 3: Structure of a soft envelope audio encoder

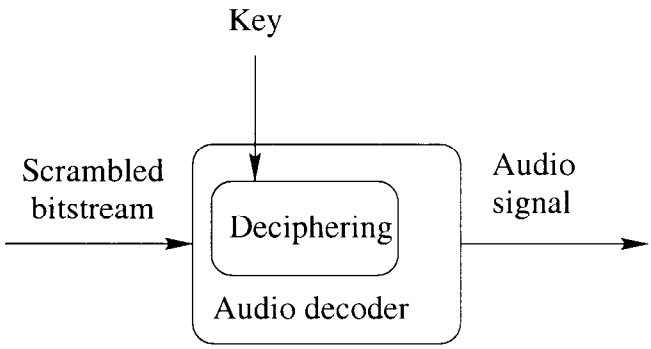


Figure 4: Structure of a soft envelope audio decoder

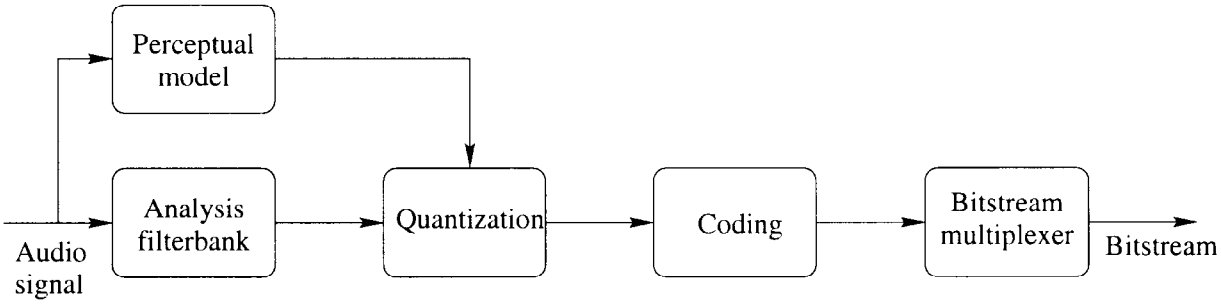


Figure 5: Generic structure of a perceptual audio encoder

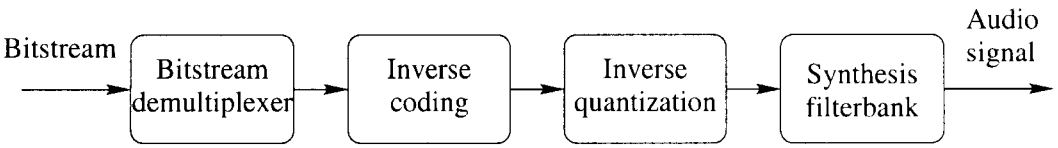


Figure 6: Generic structure of a perceptual audio decoder

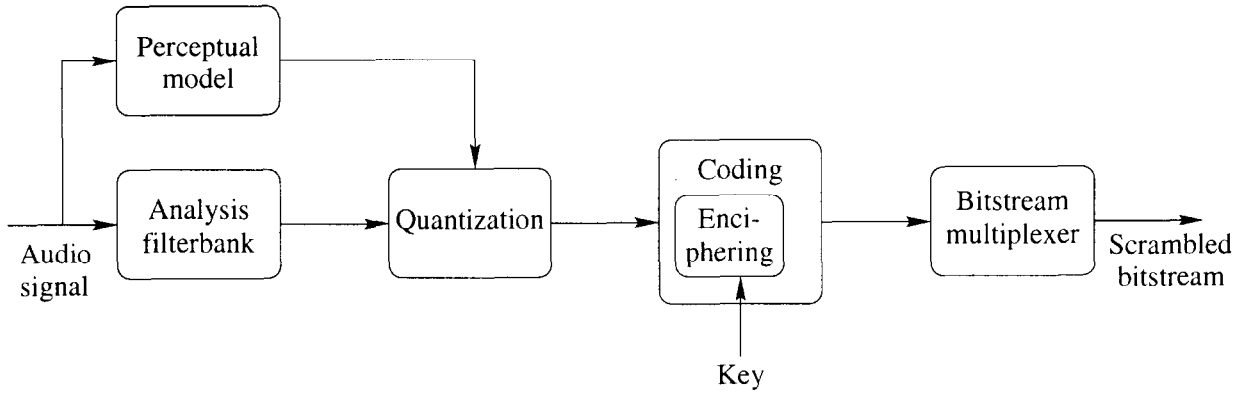


Figure 7: Structure of the compatible bitstream scrambling encoder

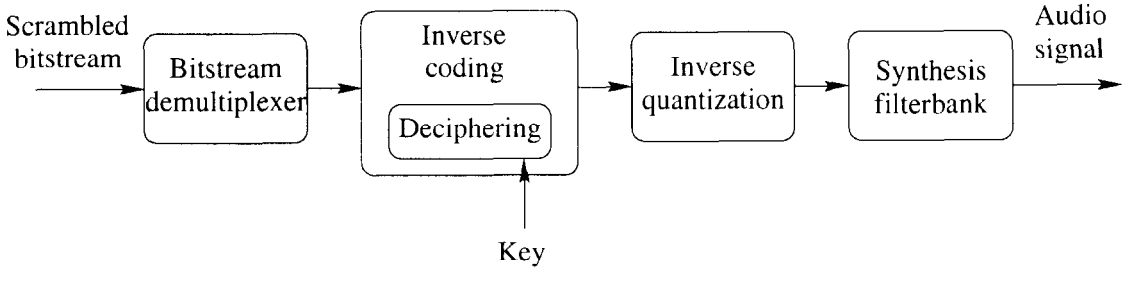


Figure 8: Structure of the compatible bitstream scrambling decoder

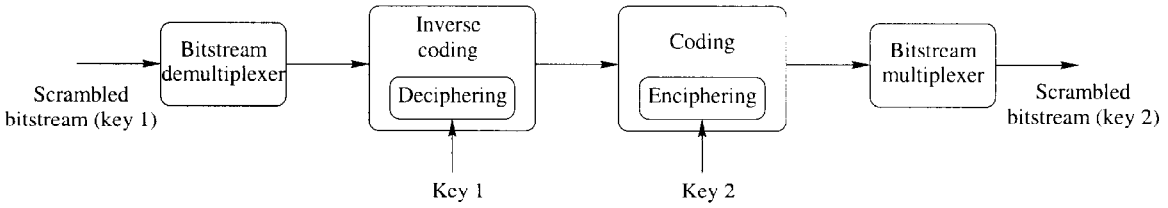


Figure 9: Structure of the compatible bitstream scrambling transcoder

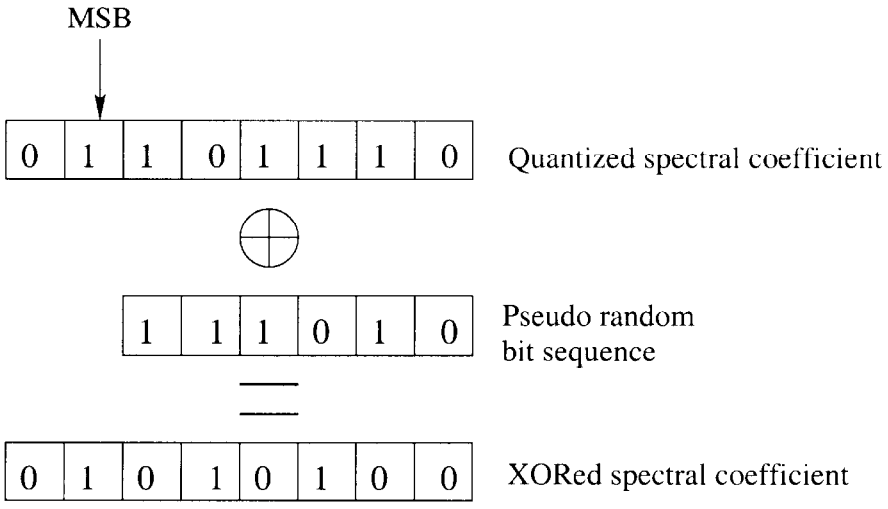
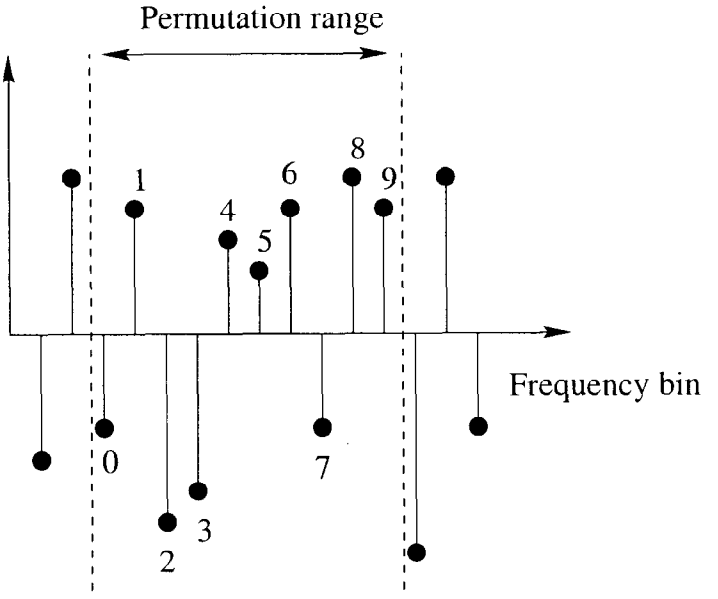
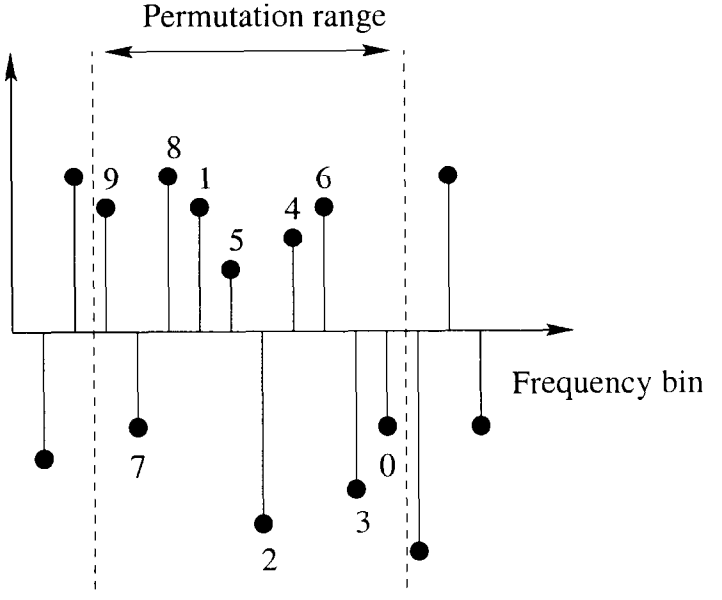


Figure 10: Scrambling through XOR operation



a) Original sequence of spectral coefficients



b) Sequence of spectral coefficients after permutation

Figure 11: Permutation of a group of spectral coefficients: a) original sequence, b) sequence after reordering (the coefficients are labelled with their original indices)