Detection of genuine lossless audio files. Application to the MPEG-AAC codec.

Olivier Derrien

Detection of genuine lossless audio files. Application to the MPEG-AAC codec.

OLIVIER DERRIEN
derrien@prism.cnrs.fr

Université de Toulon / Aix-Marseille Université / CNRS PRISM, Marseille, France

In this study, we describe an algorithm that discriminates *transcoded* from *true lossless* audio files, i.e. file that are obtained by directly ripping a genuine audio CD, from file that are decoded from a lossy audio format (e.g. MP3, AAC) and re-encoded with a *lossless* format. This method should allow both consumers and online music dealers to check the authenticity of *lossless* music files. This implementation focuses on the MPEG AAC codec, but this method can also be applied to MP3 codec. It is based on the detection of the quantization error in the time-frequency domain, without machine learning. The final results on a large database of 1576 audio files, original and transcoded with AAC from iTunes, show null false positive ratios and very low false negative ratios, possibly null for high-precision settings. We finally propose a setting which appears to be a good tradeoff between precision and execution time.

0 INTRODUCTION

In the early 2000s, the audio coding community anticipated on the massive increase of storage and bandwidth capacities by developing lossless compression algorithms as a high-quality alternative to lossy codecs like MP3 [1] or AAC [2]. In 2003, the Free Lossless audio codec (FLAC) was introduced by the Xiph organization [3]. In 2004, the Apple Lossless Codec (aka ALAC) was proposed in the iTunes software [4]. And in 2006, the MPEG-4 Audio Lossless Coder (ALS) was published in ISO/IEC 14496-3 [5]. These codecs are now widely used, especially by online music dealers who sell *CD quality* or *Studio Master* audio tracks.

However, the use of a lossless format does not guarantee that the audio is *genuine lossless*, i.e. that the audio content was not previously encoded using a lossy format. Practically, as exemplified on figure 1, there is a major difference between a audio file that was ripped from a CD (or copied from a studio master) and directly encoded using a lossless format (case A), and another file that was first encoded using a lossy format (e.g. MP3 or AAC) and *transcoded* to a lossless format (case B). If sold by a music dealer as a genuine lossless track, the *transcoded* case would clearly be a fraud. Thus, it would be of great interest for both the consumer and the music dealer to have an automatic and reliable detection tool for that purpose.

Discriminating the *transcoded* case from the *genuine lossless* case is equivalent to discriminating a decoded lossy audio file from the unprocessed file. This task can be performed by many listeners at medium/low bitrate, and only to experts at higher bitrate. However, it requires to hear the lossy file and the unprocessed file one just after another. In our use case, only one version of the audio file can be tested. Thus, there is another requirement for such a detection tool: it must operate in full blind mode, which is hardly accessible to human beings. To our knowledge, the first and only serious attempt to realize such an analyzer is the *CD Authenticity Detector* included in the *Tau Analyzer* software, released in 2005 [6]. This is a free software that can be downloaded from the internet, and it is not specific to a given lossy codec. The authors claim that this method has a a 7.5% false positive rate and a 0.6% false negative rate. This corresponds to the following standard measures: Precision = 92.5%, Recall = 99.4% and F-measure = 95.6%. However, *CD Authenticity Detector* can not be applied directly to an audio file. It treats only physical CDs. Thus, a proper evaluation of this method would require to burn *transcoded* versions of each CD on a physical discs.

In 2015, a new detection method for detecting MPEG-AAC transcoded files was included in a beta version of the *Lossless Audio Checker* (LAC) software [7], and the first tests showed both zero false-positive and zero false-negative rate on a selection of genuine lossless and AAC-encoded files at 128, 256 and 320 kbits/s. This software was presented in [8], but the genuine lossless detection algorithm was not described. In this paper, we describe precisely an updated version of the AAC detection algorithm and test it on a wider selection of audio files from various musical genres. This method can be easily adapted for

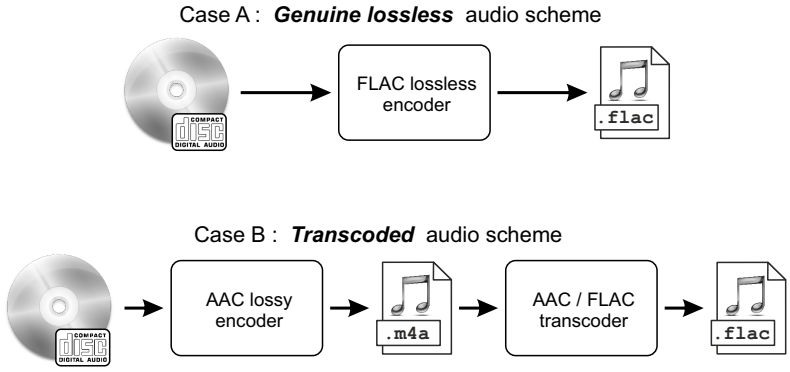

Fig. 1. Genuine lossless audio paradigm.

the MP3 case, since both codecs share the same quantization stage. In the forthcoming version of LAC, the detection will be implemented for both MP3 and AAC codecs.

This paper is organized as follows: In section 1, we describe the detection algorithm. In section 2, we explain how main parameters have been tuned, and in section 3, we evaluate the performance on a large database of audio files.

1 The detection algorithm

The distortions usually associated to lossy coding and decoding are: loss of bandwidth, pre-echo, *double-speak* effect, degradation of stereo image, reduction of dynamics, *birdies* artifacts [9]. However, there are two main objections to using such features in a detection process: These are *relative* features, which are not helpful in blind mode. And these modifications become almost imperceptible at high bitrates. This explains why a detection algorithm based on these features will probably never be reliable.

1.1 Quantization detector basics

Another way to tackle the problem is to analyze how lossy codecs transform the audio signal. The block diagram of a typical lossy codec is presented on figure 2. In the coder, the audio signal is first transformed in a time-frequency domain, e.g. using a Modified Discrete Cosine Transform (MDCT) [10] like in AAC. Then, the transform coefficients are quantized, and finally the quantization index are lossless binary coded. In the decoder, the reverse scheme is applied. The time-frequency transform and the binary coding steps are invertible, which means that they do not alter the information. In most audio codec, the quantization step consists in first scaling the transform coefficients, and then applying a rounding function. Scaling is invertible, but rounding is not. The reverse quantization reduces to the inverse scaling process. Thus, the quantization process is globally not invertible. The final distortion is caused by the rounding noise. A good way to detect a lossy coding process is to search for traces of quantization noise in the transform domain.

This problem is a difficult case, because the unprocessed audio signal is already quantized: Each sample is quantized on a uniform scale using 16, 24 ou 32 bits, depending on the audio format. Thus, the goal is to distinguish between two different quantization schemes. At high bitrate, this can be very challenging because the global amount of quantization noise can be similar in both cases. Only the structure of the distortion is different.

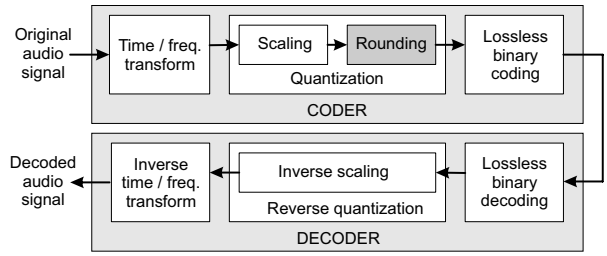

Fig. 2. Block diagram of a typical lossy audio codec (e.g. MP3 or AAC.

We propose a quantization noise detector which is basically structured as depicted on figure 3. The audio signal under test (which can be *genuine lossless* or *transcoded* audio), comes trough a time-frequency transform, scaling and rounding steps that are similar to the one used in the codec that we want to detect. Finally, we keep the difference between the input and the output of the rounding process, i.e. the rounding error.

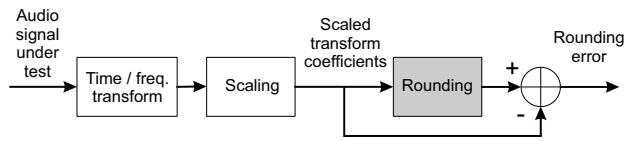

Fig. 3. Block diagram of the proposed quantization noise detector.

In the *transcoded* case, the transcoder is fundamentally a lossy decoder followed by a lossless coder. Assuming that the time-frequency transform and the scaling steps used in the detector *exactly match* those of the lossy codec, the global signal processing chain (coder-transcoder-detector) is equivalent to the one presented on figure 4. This scheme can be obtained by first ignoring the lossless encoder inside the transcoder (because it is transparent), and then remov-

ing the successive direct/inverse block pairs that do not alter the information. We get only the time-frequency transform and the quantization from the lossy coder, and the rounding function from the detector. The distinctive property of a rounding function is that, when applied twice, the second application is equivalent to the identity. Thus, the rounding error at the output of the detector should be zero. This criterion theoretically allows to discriminate the *true lossless* case from the *transcoded* case.

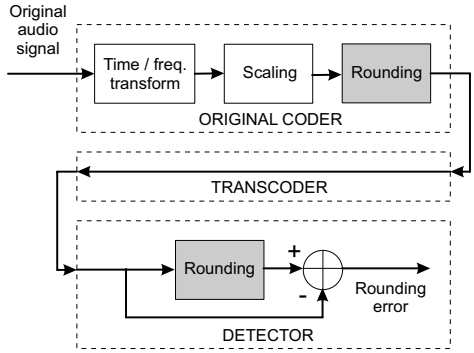

Fig. 4. Global signal processing chain in case of transcoded audio.

However, there are two major objections to this method:

1. The perfect synchronization between the coder and the detector is hard to obtain. First, because the audio signal may have been truncated and/or scaled during transcoding. Second, because time-frequency transform and scaling operations are characterized by a large set of parameters that are *a priori* unknown in the detector. One way would be to test all possible parameter sets, but the huge combinatory leads to a non tractable process. We explain in section 1.2 how to circumvent these problems.
2. The simplifications that were made wile finding the equivalent signal processing chain in figure 4 are not totally justified. Mainly because the audio at the output of the transcoder is re-quantized according to the lossless audio format. Although the resulting quantization error is usually very small, this prevents from getting a zero rounding error at the output of the detector, even when the original coder and the detector are perfectly synchronized. However, there is still a noticeable difference between both situation that allows a successful discrimination. This will be explained in section 1.3.

1.2 Coding parameters and reduction of complexity

In this section, we formalize the coding and decoding process in order to specify the coding parameters that interfere with the detection process. We assume the MPEG AAC codec, but we recall that the MP3 case would be similar.

In the monophonic case (1 channel), the input signal in analyzed by a MDCT switched between two time-frequency resolutions: long window (1024 bands) and short window (128 bands). One coding frame corresponds either to one long window (or one transition window), either to 8 short windows. In this implementation, we consider only sine windows, which seem to be the most common case in AAC implementations. We checked that encoding with Kaiser-Bessel windows does not alter the detection performance in a significant way.

In the current analysis window, the MDCT coefficients corresponding to the audio signal are noted $X(k)$ where k is the MDCT bin index (i.e. a frequency index). The scaling process consists in a gain followed by a power-law transformation:

$$|X_{sc}(k)| = \left(|X(k)|\ 2^{-\frac{\varphi_s}{4}}\right)^{\frac{3}{4}}$$

The sign of MDCT coefficients is not modified by the quantization process, and is processed separately. The scaling process is parametrized by a set of *scalefactors* noted here φ_s, which are defined as integers (positive or negative). The set of index k is split into variable-width frequency-bands, corresponding to perceptual frequency bands, and MDCT coefficients in each band are scaled unsing a single scalefactor. s is the subband index.

One can notice that a gain applied to the audio signal corresponds to a translation on scalefactors, but not necessarily by an integer value. Thus, in order to consider all possible cases, we extend the possibilities to non-integer scalefactor values in the detector.

It is important to notice that practical scalefactor values are bounded by an upper value, noted here φ_s^{dz}, which corresponds to the *dead zone*, i.e. all scaled coefficients are lower than 0.5, and will be rounded to 0. In fact, the limit value is not allways 0.5, depending on the implementation of the encoder, but 0.5 is the lower bound. In can be easily proved that:

$$\varphi_s^{dz} = \frac{16}{3} + 4\ \log_2\left(\max_s |X(k)|\right)$$

This upper bound can be estimated by observing the MDCT coefficients before quantization. We found out that best detection results were obtained by restraining the scalefactor values to $[\varphi_s^{\min}, \varphi_s^{\max}]$, where $\varphi_s^{\min}$ and $\varphi_s^{\max}$ are set respectively to $0.3\varphi_s^{dz}$ and $0.7\varphi_s^{dz}$. We use this setting in the present study, but this point will probably be finely tuned in a next version.

Then, scaled MDCT coefficients are rounded to integer values using a function noted here Rd. We get quantization index, noted here I_k:

$$I_k = \mathrm{Rd}(|X_{sc}(k)|)$$

Quantization index $I(k)$ and scalefactors φ_s are lossless coded using Huffman codebooks.

In the stereo case, 3 modes are possible:

1. Double mono: Each channel is processed like a mono signals.
2. MS stereo: Channels Left and Right are transformed to Middle (half left plus right) and Sides (half left minus sides), and each is processed like a mono signal.

3. Intensity stereo: Only the Middle channel is processed like a mono signal. The stereo information is approximated using a parametric coding scheme.

Thus, perfect synchronization between the coder and the detector requires that:

1. Both MDCT analysis start at the same time-sample. We propose to test all offsets between 0 and 1023 samples at the beginning of the audio file, and keep the most significant measure.
2. The same MDCT resolution is used in each coding frame. We propose to test short window and long window (plus the two transition windows) in parallel for each frame, and keep the most significant measure.
3. The same scalefactors are used in each window, and if a gain is applied, possibly translated. Obviously, this condition is impossible to fulfill because it would mean to test an infinite set of configurations. We propose a uniform sampling of scalefactors between $\varphi_s^{\min}$ and $\varphi_s^{\max}$.
4. In stereo, the same stereo coding mode is used. We propose to test Left, Right, Middle and Sides channels for each frame and keep the most significant measure. This includes the Intensity stereo case, where only the Middle channel will produce a significant result.

It is possible to reduce the complexity by performing a subsampling of coding frames in the audio file, and keep the most significant result. However, we experienced that the detection can be improved by selecting only high-energy frames.

Keeping the most significant measure is related to the statistical detection criterion described in the next section. Practically, this means keeping the maximum value of the detection criterion.

1.3 The statistical detection criterion

As explained in the previous section, even when the coder and the detector are perfectly synchronized, the quantization error at the output of the detector will never be exactly equal to zero. Thus, we propose a statistical detection measure that allows a robust classification between the *true lossless* and the *transcoded* case.

In the current MDCT window, we consider the energy of the quantization error in each subband s, characterized by its lower and upper frequency indexes $k_{\min}(s)$ and $k_{\max}(s)$:

$$\varepsilon(k) = \mathrm{Rd}(|X_{sc}(k)|) - |X_{sc}(k)|$$

$$E(s) = \sum_{k=k_{\min}(s)}^{k_{\max}(s)} \varepsilon^2(k)$$

Assuming that the observed signal is *genuine lossless*, $\varepsilon(k)$ can be seen as the standard quantization error at the output of a uniform unitary scalar quantizer. It can be modeled by a random variable uniformly distributed in $\left[-\frac{1}{2},\frac{1}{2}\right]$ [11]. Then, assuming that the subband is large enough, according to the central-limit theorem, $E(s)$ can be modeled by a Gaussian random variable of mean and variance:

$$\mu = \frac{K(s)}{12}, \qquad \sigma^2 = \frac{K(s)}{180}$$

where $K(s) = k_{\max}(s) - k_{\min}(s)$ is the subband width (in MDCT bins).

When the observed signal is *transcoded*, the probability distribution function (pdf) of $E(s)$ is unknown, but it support is condensed around 0.

Our goal is to make the difference bewteen two pdfs which typical shapes are exemplified on figure 5. A good way to do so is to set a threshold τ between 0 and μ, and estimate the probability that $E(s)$ is lower than τ. In the *true lossless* case, this probability can be written as:

$$\mathrm{proba}\{E(s) < \tau\} = \frac{1}{2}\left[\mathrm{erf}\left(\frac{\mu}{\sqrt{2}\,\sigma}\right) - \mathrm{erf}\left(\frac{\mu-\sigma}{\sqrt{2}\,\sigma}\right)\right]$$

where erf is the standard error function [12]. As we can see on figure 5, in the *transcoded* case, the probability is higher.

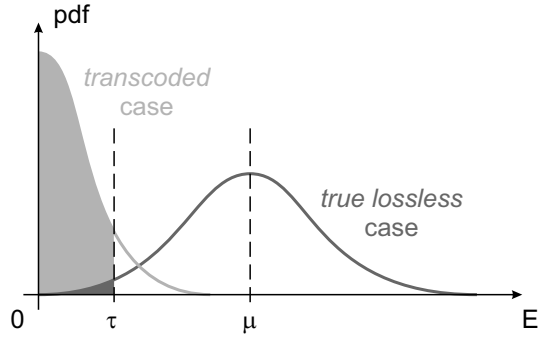

Fig. 5. Typical probability density function of the rounding error. Probability that $E(s) < \tau$ is plotted as a surface.

Practically, the probability is estimated by counting the occurrences of $E(s) < \tau$ along subbands. In order to get consistent measures, we set a different threshold τ in each subband, such that the theoretical probability in the *true lossless* case is constant along subbands. Then, we can sum the occurrences along subbands in the current frame, which improves the robustness of the detection process (equivalent to averaging the detection over subbands). The final detection criterion is the total occurrence count normalized by the number of subbands, which can be interpreted as a likelihood measure for the *transcoded* case.

The formula that sets $\tau(s)$ as a function of the target probability P and the subband width $K(s)$ is:

$$\tau(s) = \frac{K(s)}{12} - \sqrt{\frac{K(s)}{90}}\,\mathrm{erfinv}\left(\mathrm{erf}\left(\frac{\sqrt{90K(s)}}{12}\right) - 2P\right)$$

where erfinv is the inverse standard error function [12]. Here, we set $P = 0.01$. The thresholds $\tau(s)$ are precomputed and stored, to save computation time.

When the difference between the scalefactor values used in the coder and in the detector increases, the pdf in the *transcoded* case distorts to take progressively the Gaussian shape of the *true lossless* case. But since this distortion is

progressive, it allows a true positive detection even when the scalefactor values do not match. This phenomenon is exemplified on figure 6. We consider two versions of a real audio file: One unprocessed, and another encoded and decoded using a modified AAC, such that one particular subband ($s = 22$, $K(s) = 16$) uses the same scalefactor value for all frames (long window only). We apply the detector to these two signals, and observe only this particular subband. We compute the histogram of the rounding error by aggregating all frames. This histogram is an estimation of the error pdf. We also plot the threshold used for the statistical detection process. On the left plot, the detector uses the same scalefactor value as the encoder. The contrast between both situations is clear. On the right plot, the detector scalefactor value is increased by 1 unit. The pdf in the *transcoded* case resembles the pdf in the *true lossless* case, but the probability measure is still discriminant. Note that this experiment was possible only because we modified the AAC encoder such that we can aggregate the results of all frames. In the real world, the number of MDCT coefficients in each subband and each frame is much too low to estimate accurately the pdfs.

1.4 Description of the algorithm

The detection algorithm can be summarized as follows.

Pre-compute probability thresholds $\tau(s)$
Extract L,R,M,S channels
$L = 0$
for offset = 0 : 1023
  Apply offset
  Segment audio data in frames
  Sort frames by decreasing energy in M channel
  for $i_f = 1 : N_f$
    for channel = L,R,M,S
      Apply MDCT long and short to frame # i_f
      for $s = 1 : N_s$
        Compute φ_s^{dz}, $\varphi_s^{\min}$ and $\varphi_s^{\max}$
        $c = 0$
        for $i_{sf} = 0 : N_{sf} - 1$
          $\varphi_s = \varphi_s^{\min} + i_{sf}(\varphi_s^{\max} - \varphi_s^{\min})/(N_{sf} - 1)$
          Scale MDCT coefficients using φ_s
          Apply rounding and compute error
          Compute error energy E in s
          if $E < \tau(s)$, $c = c + 1$
        end for
        $c = c/N_s$
        $L = \max(L, c)$
      end for
    end for
  end for
end for
if $L > \lambda \Rightarrow$ *transcoded*
else $\Rightarrow$ *true lossless*

L stands for the likelihood of the *transcoded* case. N_f, N_{sf} adn N_s are respectively the number of frames and scalefactor values used in sampling, and the number of subbands (defined in the AAC standard). i_f, i_{sf} and s are respectively frame, scalefactor and subband indexes. c is counts the occurrences of $E(s) < \tau(s)$ in the current frame. λ is the threshold that allows to take the final decision (not to be confused with the probability thresholds $\tau(s)$).

2 Setting the sampling parameters

In the algorithm described in section 1.4, three parameters were not specified: N_f, N_{sf} and λ. In this section, we describe an experiment that allows to accurately set the relative values for N_f and N_{sf}.

It is quite intuitive that the higher N_f and N_{sf}, the more reliable is the detection, but also the higher is the computation time. Thus, a tradeoff must be found. We organized an experiment with a small set of 10 stereo audio files from different genres ripped from original CDs. In the *transcoded* case, we used an AAC codec operating at 128, 256 and 320 kbits/s. We use the codec from iTunes software, LC profile, automatic stereo mode. Note that these bitrates values can be considered as high, since the default setting is 128 kbits/s.

On figure 7, we plot the log-likelihood of the *transcoded* case, $\log_{10}(L)$, for $N_f \times N_{sf} = 64$, for two special files: An "easy" case: "Cocaine" by J.J. Cale (blues rock), and a "difficult" case: "Goldberg Variation" by J.S. Bach (performed by G. Gould on piano). The description of the small database and the full result set it available on the companion webpage: `https://potion.prism.cnrs.fr/JAES2018.html`.

We expect a near-constant likelihood value in the *true lossless* case, corresponding to the left-side tail of the gaussian curve plotted on figures 5 and 6. It appears that the likelihood exhibits few variations, between $10^{-1.8} \approx 0.016$ and $10^{-1.6} \approx 0.025$. Globally, the symmetric setting 8x8 ($N_f = 8$, $N_{sf} = 8$) gives the lowest likelihood value.

In the *transcoded* case, likelihood values are almost constant across the settings, except for the last one ($N_f = 64$, $N_{sf} = 1$) where the likelihood value significantly falls down. On "Goldberg Variation", the *true lossless* and the *transcoded* cases can not be discriminated with this setting. Obviously, choosing only one scalefactor value in each subband is not enough. For all other settings, the *true lossless* and the *transcoded* cases can be efficiently discriminated by simple thresholding, with a threshold value between $10^{-1.6} \approx 0.025$ and $10^{-1.5} \approx 0.032$. The final threshold values will be determined in the next section.

It appears that detection is easier for low bitrates than for high bitrates, which is easy to understand: Higher bitrates generate lower quantization error level, and thus quantization error is more difficult to detect. The detection is also easier with "Cocaine" than with "Goldberg Variation". This correlates with observations that were made during the validation tests of all lossy codecs: Signals with sharp transients and wide bandwith exhibit higher subjective distortion levels, i.e. are compressed in a less efficient way. "Goldberg Variation" is a very slow piece of piano, with no sharp attacks, and a limited bandwidth, and thus can be more easily compressed.

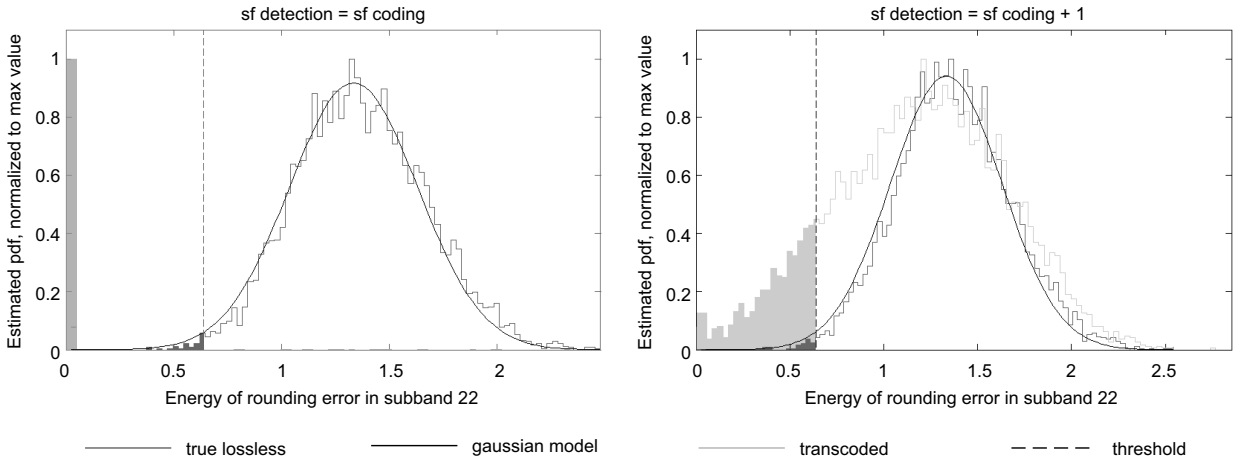

Fig. 6. Rounding error pdfs in subband 22 when scalefactors match (left), and when the detection scalefactor is one unit higher than the coding scalefactor(right). Probability that $E(s) < \tau$ is plotted as a colored surface.

Finally, symmetric settings ($N_f = N_{sf}$) appear to be the best choice because these give the highest margin between the *true lossless* and the *transcoded* case.

3 Setting the detection threshold and global performance evaluation

Setting accurate values for the detection threshold λ require a large database of audio signal, form various genres.

Our large database was obtained by ripping 100 genuine commercial CDs. Double or triple CDs were counted as 1. We finally get 1576 audio files in PCM format, each declined in 4 versions: unprocessed (for the *true lossless* case), coded and decoded with AAC at 128 kbits/s, at 256 kbits/s and at 320 kbits/s (for the *transcoded* case). We used the AAC codec from iTune, LC profile, automatic stereo mode.

We consider the following settings: $N_f \times N_{sf} =$ 8x8, 16x16, 32x32 and 64x64. For each setting, we run the detector on each 4 versions of each files, and store the likelihood values L. For each setting, the detection threshold λ is set to the minimum value such that the *false positive ratio is null for all bitrates*, i.e. no unprocessed file is classified as *transcoded*. We get $\lambda = 0.031$ for the 8x8 setting, $\lambda = 0.019$ for the 16x16 setting, $\lambda = 0.0145$ for the 32x32 setting and $\lambda = 0.0125$ for the 64x64 setting.

In table 1, we give the final Recall and F-measure values for each setting and each bitrate. The Precision is allways 100%. It appears that the F-measure values are much better that the ones obtained with *CD Authenticity Detector*. As expected, the F-measure improves when N_f and N_{sf} increase, and when bitrate decreases. The classification is perfect for $N_f \times N_{sf} =$ 64x64. However, we can not guarantee that is would still be the case on any audio file and with any AAC codec.

The description of the large database and the detailed results for each CD are available on the companion webpage: `https://potion.prism.cnrs.fr/JAES2018.html`.

Setting	Bitrate	Recall	F-measure
8x8	128 kbits/s	100%	100%
8x8	256 kbits/s	98.10%	99.04%
8x8	320 kbits/s	96.13%	98.03%
16x16	128 kbits/s	100%	100%
16x16	256 kbits/s	99.56%	99.78%
16x16	320 kbits/s	98.86%	99.43%
32x32	128 kbits/s	100%	100%
32x32	256 kbits/s	99.94%	99.97%
32x32	320 kbits/s	99.68%	99.84%
64x64	128 kbits/s	100%	100%
64x64	256 kbits/s	100%	100%
64x64	320 kbits/s	100%	100%

Table 1. Recall and F-measure for the final classification task.

However, a low computation time also is highly desirable, considering that most users wish to analyze large audio libraries. In table 2, we give the execution times corresponding to the analysis of one stereo audio file using our public Matlab 8 implementation, on an Intel-G7 desktop running with Windows 7, with and without the parallel computing toolbox (offset values distributed over 4 workers). Note that much faster implementations can be obtained using compiled languages.

Setting	Serial process	Parallel process
8x8	2 min 40 s	1 min
16x16	10 min 20 s	3 min 40 s
32x32	40 min	13 min 20 s
64x64	154 min 10 s	50 min

Table 2. Execution times for analyzing one stereo audio file, Matlab 8 implementation.

One can see that increasing N_f and N_{sf} dramatically increases the computation time. The 64x64 setting, which produces a perfect classification on our large database, is

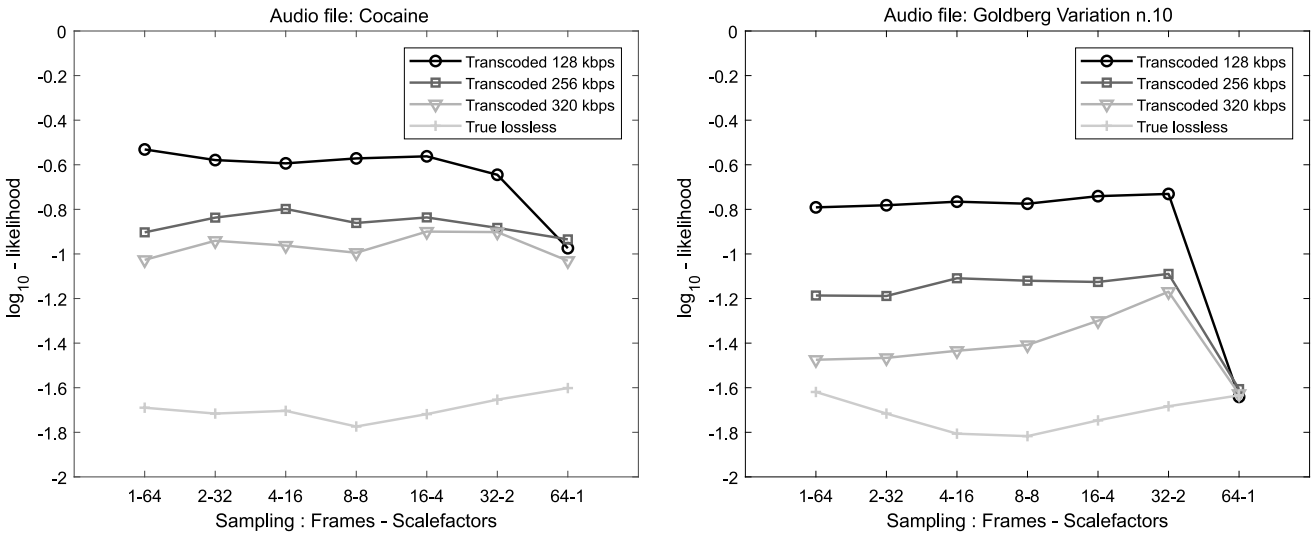

Fig. 7. Log-likelihood measure of the *transcoded* case, for two audio files and for different settings N_f - N_{sf}.

obviously not tractable. Thus, the 8x8 setting appears to be a good tradeoff between precision and speed.

4 CONCLUSIONS

In this study, we described an algorithm that allows to discriminate *transcoded* from *true lossless* audio files. This implementation focuses on the MPEG AAC codec, but the same method can be applied to the MP3 codec. This requires to replace the MDCT analysis filterbank with the PQMF/MDCT filterbank used in MP3. The remaining of the algorithm is the same, since both codecs use the same quantization scheme.

Our method is based on the detection of the quantization error distribution in the time-frequency domain, without machine learning. The final results on a large database of 1576 audio files, original and transcoded with the AAC codec from iTunes, show very high F-measure values, better than *CD Authenticity Detector*.

We show that a perfectly accurate classification can be reached, provided a sufficiently long computation time. We also recommend a standard setting (8x8) which appears to be a good tradeoff between precision and speed. Furthermore, our method is naturally immune to truncation of audio files and global gain modification.

This method was implemented in the *Lossless Audio Checker* (LAC) software, both for AAC and MP3 codecs. This should allow both consumers and online music dealers to check the authenticity of *lossless* music files.

5 REFERENCES

[1] ISO/IEC, "11172-3: Information Technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s, Part 3: Audio," ISO, Tech. Rep. (1993).

[2] ——, "13818-7: Generic Coding of Moving Pictures and Associated Audio: Advanced Audio Coding," ISO, Tech. Rep. (1997).

[3] J. Coalson, "Free lossless audio coder (FLAC), detailed format description," available online at `http://www.xiph.org/flac/format.html`

[4] "Apple lossless," available online at `http://www.applelossless.com`

[5] ISO/IEC, "14496-3:2005/Amd2:2006, Audio Lossless Coding (ALS), new audio profiles and BSAC extensions," ISO, Tech. Rep. (2006).

[6] "Cd authenticity detector," available online at `http://tausoft.org/wiki/Main_Page`

[7] "Lossless Audio Checker," available online at `http://losslessaudiochecker.com`

[8] J. Lacroix, Y. Prime, A. Remy, and O. Derrien, "Lossless audio checker: A software for the detection of upscaling, upsampling and transcoding in lossless musical tracks," in *139th Convention of the AES*, New York, USA (October 2015).

[9] I. Corbett, "What data compression does to your music," *Sound On Sound* (April 2012).

[10] J.P. Princen and A.B. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," *IEEE tr. on Acoustics, Speech and Signal Processing*, vol. ASSP-34, pp. 1153–1161 (1986).

[11] S. Lipshitz, R. Wannamaker, and J. Vanderkooy, "Quantization and dither : a theoretical survey," *Journal of the Audio Engeneering Society*, vol. 40, no. 5, pp. 355–374 (May 1992).

[12] M. Abramowitz and A. Stegun, *Handbook of Mathematical Functions*. New York: Dover Publications Inc. (1970).

THE AUTHOR

Olivier Derrien

Olivier Derrien was born in Aix-en-Provence, France, in 1974. He recieved the State Engineering degree in 1998 and the Ph.D. degree in audio processing in 2002, both from Télécom ParisTech (formerly ENST), Paris, France. From 2002 to 2003, he was a Teaching and Research assistant at the University of Paris-XI, Orsay, France. He joined the University of Toulon, Toulon, France, in 2003 as an Associate Professor in the field of telecommunications. In 2008, he joined the Laboratory of Mechanics and Acoustics (CNRS-LMA), Marseille, France, as an Associate Researcher. In 2017, he moved to the Perception, Representation, Image, Signal and Music (CNRS-PRISM), Marseille, France. His research interests include audio signal processing, especially audio coding, audio synthesis for virtual reality, music synthesis and audio effects. He is a co-author of 7 journal papers and 20 international conference papers.