Audio Engineering Considerations for a Modern Mixnet

Brett Alexander Preston

October 11, 2023

This work is supported by a grant from the Wau Holland Foundation.

Privacy-enhancing technologies have always faced a challenge in balancing security guarantees with user experience that would bring people to the service. In the contemporary communication landscape, most users expect their messengers to allow for some form of audio communication. We would therefore like to meet that demand without compromising anonymity.

The most ambitious private messengers being built today are those based on modern Mixnet designs. They introduce padding, in addition to a network of relays, and have to contend with latency. They distinguish themselves by considering realistic powerful adversaries and endeavoring to protect both the content of the communication and the metadata. This turns out to be crucial in the case of audio communication, since in most common implementations the metadata leaks content, so you can't have one without the other. We will therefore discuss implementation recommendations from the point of view of audio engineering of adding audio communication to a Mixnet. We will look at how we can balance efficiency and user experience in this unique setting while upholding security guarantees.

Contents

1 Content leaks in common encrypted VoIP implementations **2**

2 Recommendations for audio encoding and decoding **2**

2.1 Audio codecs 3

2.2 Bitrate 4

2.3 Bandwidth 4

2.4 Frame size 4

2.5 Algorithm complexity 4

3 Signal processing, noise reduction and equalizing **5**

4 Recommendations summary **5**

A modern Mixnet typically uses padding in order to mitigate traffic analysis. [1] This means that a constant amount of traffic is being sent by a user at all times. The amount of traffic it allows for is also an upper limit to how much data a user can send. It is not realistic to set this limit too high, since it would quickly add up to a significant drain on a user's resources. Therefore we should expect any audio communication to either have to be compressed to a low bitrate, or take a long time to travel through the network, which can make real-time audio communication impossible. Therefore we will consider non-synchronous push-to-talk messaging as an important mode of audio communication in this setting.

1 Content leaks in common encrypted VoIP implementations

It has been demonstrated that encrypted real-time VoIP communications can produce devastating data leaks by not accounting for the fact that different phonemes are connected to different bitrates. Already in 2011, researchers were able to reconstruct sections of conversations from encrypted connections based on packet traffic patterns alone. [2] Today, the threat is even more dire due to increased prevalence of Machine Learning and therefore the automation of powerful statistical analysis. [3] And yet popular communication software does not address this.

One can typically mitigate this problem by either using constant bit-rate encoding, therefore increasing the overall file size, or opting for push-to-talk messaging instead of real-time communication, where you process the entire recording. This goes back to the Anonymity Trilemma [4]: to maintain security guarantees, you have to compromise either on the overhead or on latency. But a Mixnet with padding has already made these compromises and so it faces these challenges by design.

2 Recommendations for audio encoding and decoding

Encoding and decoding audio in a Mixnet has a unique set of challenges. We are particularly interested in efficiency, as there may be a strict bandwidth limit, but at the same time we have access to the computing power of a modern device and modern audio encoding and decoding tricks. We can reasonably expect to deal with some latency, and depending on the transport-layer protocol used we may have to consider some packet loss. We can also expect a modern Mixnet to prioritize security and to require its components to use licensing that supports freedom.

We will assume that padding and/or non-synchronicity allow for an implementation of VBR encoding without leaking content. We should still keep in mind security risks that come from haphazard implementations of VBR decoding, as sometimes they might allow for injection of malicious code. We should make sure the decoder we're using has been audited. An example of a decoder written with security in mind is Rust-based Symphonia [5].

The following table is a comparison of file sizes in kBs generated by VBR encoding in various codecs, starting with a lossless wav file. The cells are clickable, so the reader can verify the sound quality. For the HTML version of this table, visit https://brettpreston.github.io/mixnet-samples.

Codec	Bitrate	Band width	Complexity	Deep voice (32sec)	Deep voice + noise 35 sec	High Voice	High voice+noise 49 sec	Generic rock 51 sec
wav				2930	2990	4560	4330	6620
opus	64 kbps	wide	10	304	271	456	390	293
opus	32 kbps	wide	10	113	131	170	190	137
opus	16 kbps	wide	10	59.1	68.5	89.7	98.4	71.6
opus	12 kbps	wide	10	45.4	51.8	69.4	74.7	54.5
opus	12 kbps	wide	5	45	50.4	68.4	72.5	53.5
opus	12 kbps	narrow	5	49	54.5	76.2	78.6	60.2
opus	6 kbps	wide	10	25.9	26.9	41	39.3	30.6
opus	6 kbps	wide	5	25.6	26.4	40.1	38.6	30.2
opus	6 kbps	narrow	5	25.4	28.2	36.4	40.7	31.2
speex	64 kbps	wide	10	165	166	256	242	126
speex	32 kbps	wide	10	95.2	109	145	158	108
speex	12 kbps	wide	10	36.3	42.4	55.7	51.1	54.3
speex	4 kbps	wide	10	22.5	25	34.9	36	35.3
codec2	3200bps	default	n/a	12.7	14.1	19.9	20.4	15.5
codec2	1200bps	default	n/a	4.74	f	7.47	f	f
codec2	450bps	default	n/a	2.4	f	3.73	f	f

The key takeaways from this table can be summarized as follows. Opus delivers the best quality of the three codecs, and better file size than Speex, and is the only one that can handle more than speech. With Opus we observe diminishing returns with quality above 12 kbps. Bandwidth has a small impact on the file size. Algorithm complexity has negligible impact on the file size, it primarily impacts local processing requirements. Codec2 delivers very, very small file size but can't handle noise or music at all. We go into detail on all of these points below.

2.1 Audio codecs

We have selected audio codecs that can be candidates for use in this setting. They deliver either impressive sound quality at low bitrates, or good sound quality and impressively low bitrates, and each has different strengths. Opus and Speex are under a BSD license, and Codec2 under LGPL.

Opus and Speex

By far the most popular codec today, Opus is used in most modern VoIP systems. It is a versatile audio codec that is known for its high-quality, crisp sound reproduction, suitable for a wide range of applications, including voice communication and music streaming. It has ready implementations in many programming languages, including Go [6] and Rust [5]. It is also well documented, simplifying its potential integration into various projects. [7]

Another fine codec for speech compression is Speex, which was popular in VoIP systems before the rise of Opus. It delivers clear and bright speech, but is not meant to be used for other sound. There are extensive resources which compare Opus to Speex [8] [9], and it is clear that Opus is both more efficient and more versatile.

Codec2

For this special use case of Mixnets, we may also consider Codec2 because it is capable of extremely efficient encoding. Its efficiency relies heavily on sinusoidal coding and a narrow frequency band, which means that we quickly lose clarity and some distinguishing features of the original speech recording. The simplified harmonic content encodes less information per frame compared to the popular codecs.

However, in this particular context both the extremely small file size and the voice masking provided by the loss of distinguishing features may be desirable. If we were to implement Codec2 but still wanted to improve clarity, we could consider the following adjustments:

- Implementing a noise filter before encoding. A demonstration of the effect denoising can have on Codec2 can be found in section 3 of this analysis.
- If trade-offs in the codec itself are acceptable, a wider frequency band on the higher end and improved handling of noise, both in noise reduction and encoding of consonants, would go a long way to improve the sound quality. Out-of-the-box Codec2 uses a very narrow subset of human voice frequencies and doesn't handle consonants well, which means it could have trouble with some consonant-heavy languages. These adjustments would come at the cost of compression efficiency.

  It appears that Codec2 was originally intended for radio broadcast, in which case these shortcomings would be somewhat mitigated by post processing typically used for radio broadcast.
- As it stands now, we can mitigate the losses after decoding by boosting what little of the higher frequency range survived. An equalizer is the most resource-efficient way to address this problem.
- Implementing a neural network-based decoder, akin to WaveNet, for frequency "reconstruction." [10] This is very resource intensive, and so may not feasible on most personal devices. One should also keep in mind that these tools don't reconstruct the original voice, they create a clean simulacrum of a human voice which may not sound like the original speaker.

It should be emphasized that Codec2 is unlikely to provide user experience on par with Opus and so it is only applicable in a limited set of use cases.

2.2 Bitrate

Naturally, we would like to find an optimal point where the bitrate is low and the sound quality is good. As can be heard in the samples above, as well as quantified in [9], we experience diminishing returns above 12kbps with Opus when it comes to speech compression. As long as minimizing the file size is a priority, either 12 or 16kbps appears to be a fine choice. While we are encoding speech, it also doesn't make sense to encode multiple channels.

The priorities change if we are encoding music - then higher bitrates make a big difference, as can be heard in the provided samples. If we were planning for music streaming we would also be hoping to allow for (at least) stereo. This is unlikely to come into play in our use case however, and so we will settle on 12kbps, mono encoding. The following figure comes from [9].

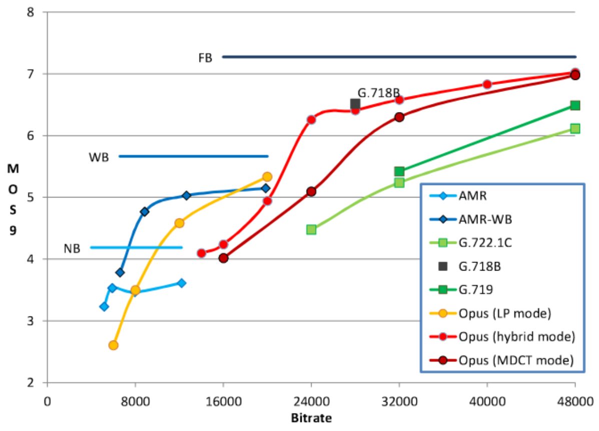

Figure 1: *Voice quality evaluation results in clean speech*

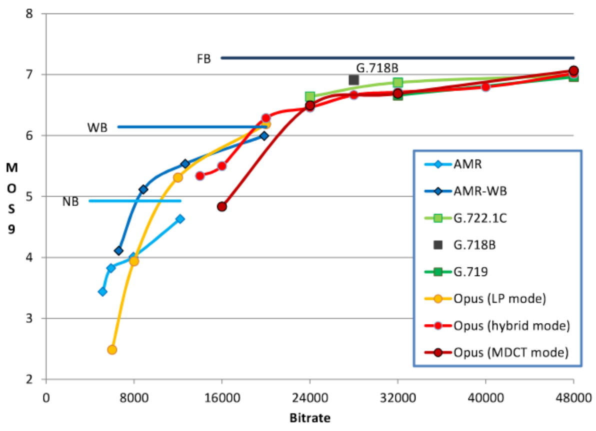

Figure 2: *Voice quality evaluation results in noisy speech*

If we choose Codec2 there is little reason to go below 3200bps, as the quality at lower bitrates is not competitive in modern VoIP landscape, and recordings with background noise or music result in a jumbled mess.

2.3 Bandwidth

We should use wide band compression. In most settings it has little impact on the file size, but is a huge gain in audio quality and clarity.

2.4 Frame size

Encoding with Opus, frames lengths under 20ms at low bit rates have audible distortions as well as frames sizes over 80ms. This demonstration uses 6VBR wide band, in order to accentuate the distortion: opus-frames. In practice, this is a lower bitrate than we would use and so the result would sound better.

2.5 Algorithm complexity

The algorithm complexity impacts the processing power required on a device more than it does the file size. In our use case, where we expect the system to be used on modern devices there is no reason not to opt for higher complexity, since it's an easy gain in quality without compromising on the resources that are scarce. The recommendation is complexity 10, unless we expect to work on older mobile devices with a real-time audio stream, in which case we may want to choose 5.

3 Signal processing, noise reduction and equalizing

Background noise tends to be an issue with voice recordings, we could implement a noise filter with a small processing footprint, as well as a dynamic range compressor/limiter before encoding the message. This will improve the likelihood of a clear and appropriately loudness balanced audio message.

Noise reduction is recommended pre-encoding for maximum clarity. State-of-the-art noise suppressors tend to be based on neural networks, such as [11] [12]. This is somewhat resource intensive, but not out the question on mobile devices. There is an analysis of XIPH's efficacy and efficiency at https://jmvalin.ca/demo/rnnoise/. A demonstration of this process can be found here here.

A potential alternative is an automated Fast Fourier Transform noise suppressor, however, that is likely to involve extensive customization of available tools.

When it comes to Codec2 at 1200bit/s to 3200bit/s, an EQ boost of several decibels at 3kHz is a simple and effective way to improve consonant clarity. Noise suppression and equalizing can make a big difference when making Codec2 viable as demonstrated here. Compare with the original sample here. Without these adjustments, Codec2 may not meet the quality expectations of today's users.

4 Recommendations summary

- Before encoding: noise suppression with XIPH or a similar plugin, or a heavily customized Fast Fourier Transform process.
- In most use cases, encoding with Opus, 12kbps VBR, mono, wide band, complexity 10, frame size 20ms.

  If processing power is scarce, complexity 5.

  For extreme file compression and a partially masked voice, Codec2 3200bps, "Natural" setting. The quality in Codec2 could be improved by widening the band on the higher end. Noise suppression is crucial.
- A security conscious decoder such as [5] or [6].
- After decoding: EQ boost of several decibels at 3kHz, especially with Codec2.

Acknowledgments

Special thanks to EJ Infeld for help with formatting and editing this analysis, as well as providing valuable context about Mixnets, and to the Wau Holland Foundation for funding this work.

References

[1] Ania Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system, 2017.

[2] Andrew M. White, Austin R. Matthews, Kevin Z. Snow, and Fabian Monrose. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks, 2011.

[3] Chenggang Wang, Sean Kennedy, Haipeng Li, King Hudson, Gowtham Atluri, Xuetao Wei, Wenhai Sun, and Boyang Wang. Fingerprinting encrypted voice traffic on smart speakers with deep learning, 07 2020.

[4] FRL. Anonymity trilemma. `https://freedom.cs.purdue.edu/projects/trilemma.html`.

[5] Philip Deljanov. Symphonia. `https://github.com/pdeljanov/Symphonia`, 2022.

[6] Steffen Vogel Egon Elbre, Sean DuBois. Pion. `https://github.com/pion/opus`, 2023.

[7] IETF. Opus technical details. `https://datatracker.ietf.org/doc/html/rfc6716`.

[8] Opus. Opus comparison. `https://www.opus-codec.org/comparison/`.

[9] Anssi Rämö and Henri Toukomaa. Voice quality characterization of ietf opus codec. In *Interspeech*, 2011.

[10] W. Bastiaan Kleijn, Felicia S. C. Lim, Alejandro Luebs, Jan Skoglund, Florian Stimberg, Quan Wang, and Thomas C. Walters. Wavenet based low rate speech coding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 676–680, 2018.

[11] XIPH. Rnnoise. `https://github.com/xiph/rnnoise`, `https://jmvalin.ca/demo/rnnoise/`.

[12] Werman. Noise suppression. `https://github.com/werman/noise-suppression-for-voice/`.