Free and Open
COMMUNICATIONS
https://foci.community on the Internet

Voiceover: Censorship-Circumventing Protocol Tunnels with Generative Modeling

Watson Jia
Princeton University

Joseph Eichenhofer
Dropbox

Liang Wang
Princeton University

Prateek Mittal
Princeton University

Free and Open Communications on the Internet 2023(1), 67-80

Voiceover: Censorship-Circumventing Protocol Tunnels with Generative Modeling

Watson Jia
Princeton University

Joseph Eichenhofer
Dropbox

Liang Wang
Princeton University

Prateek Mittal
Princeton University

Abstract

Censorship regimes are continuously adopting and deploying state-of-the-art techniques to detect and prosecute open communication on the internet. Multimedia protocol tunneling seeks to disguise covert data communication by processing it directly through a legitimate audio/video communication system. Systems like VoIP and video streaming services use variable bitrate encoding schemes, which leak characteristics of the content they carry through packet sizes and timing. In what is called a content mismatch attack, censors can distinguish between a channel carrying legitimate media content and one carrying covert data content. We address content mismatch attacks by introducing a novel traffic-shaping technique that models the normal media content and applies its properties to the covert content. We constructed a generative machine learning model to restrict covert data transmission such that its timing properties match properties learned from real two-person conversations. Our evaluation finds that modeling the timing properties in the application layer content reduces distinguishing features in the encrypted network traffic. This mitigates content mismatch attacks on coarse-grained timing properties.

1 Introduction

Censoring regimes rely on a number of techniques ranging from simple address matching to advanced machine-learning classifiers on deep packet inspection traces [1] to enable Internet censorship. In response, researchers have designed and implemented many different tools to enable access to the internet even under powerful censorship. Particularly, multimedia protocol tunneling is the most recent development towards unobservable communication [2–4]. However, these methods can be easily identified by censors [5] (section 2). In particular, censors can infer properties of covert data even when it is encoded and transmitted through another protocol.

In this paper, we present the next novel development in making indistinguishable censorship circumvention communication. Our key insight is that we can learn and model the properties of the content carried by multimedia protocol tunnels. With this technique, we reduce the mismatch between covert content and true content when transmitting through a multimedia protocol tunnel. By doing so, we reduce the distinguishing properties present in the network traffic and decrease the censor's ability to identify and censor transmissions.

Voiceover, our approach to this problem, is an audio-based multimedia protocol tunnel much like previous work. We transmit data as audio by processing it with convolutional coding and quadrature amplitude modulation. That audio is carried by a Skype audio call between two endpoints to cross a censor's border. Our novel approach to improve indistinguishability is a generative adversarial network that learns and models the distribution of conversation timing properties. This model shapes the audio that Voiceover transmits through Skype such that the timing properties of our covert protocol tunnel transmission match the timing properties of a true Skype call's audio. This mitigates a censor's ability to identify differences between Skype connections carrying normal audio and Skype connections carrying covert data.

Our key contributions are presented in this paper:

(1) We propose a novel design to mitigate content mismatch attacks in censorship circumventing protocol tunnels using generative modeling.

(2) We fully implement this design as an open-source prototype for evaluation and to facilitate future work.

(3) We evaluate the security improvement of Voiceover versus a baseline implementation similar to existing techniques. A classifier trained on network traces for the existing technique could identify the protocol tunnels with an auROC of 0.98 and an aucPR of 0.96, while a classifier trained to identify our Voiceover protocol tunnel could achieve only an auROC of 0.68 and aucPR of 0.48.

2 Background and Related Work

This section defines the threat model we use for censorship in this paper and summarizes the key points in recent work that led to the problem we address in this paper.

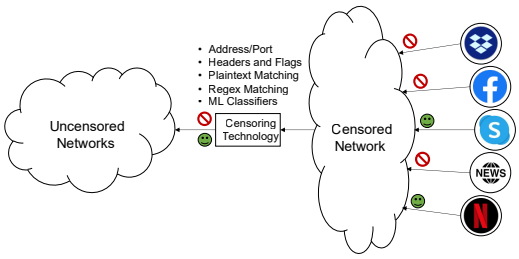

Figure 1: Threat model

Threat model. As in Figure 1 we assume that the censor can apply a wide range of techniques to detect and block censored servers and applications (e.g., anti-censorship tools). However, the censor does not block certain services for economic or political reasons, and will perform sophisticated traffic analysis to detect the "abuse" of those allowed services. However, we do not consider an adversary with unlimited computational/storage resources. Rather, we consider the adversary as a network device that is attempting to classify network flows in real time by processing packet-level features associated with a given packet flow. Unique to our work, we also assume that this adversary is memory limited, which caps the number of overall consecutive packets that can be considered at once for the purposes of flow classification. These adversary characteristics inform our evaluation of Voiceover in section 4.

Obfuscation. Early work on protocol obfuscation techniques primarily aimed to *remove* all distinguishing features from covert traffic. Examples of obfuscators include Dust and ScrambleSuit [6, 7]. Tor instantiated this idea as the obfsproxy pluggable transport and several iterations of such obfuscation proxies have since been developed, including obfs2, obfs3, obfs4, and obfs5 [8–11]. However, Wang et al. showed that obfuscation techniques such as this could be cheaply identified by entropy analysis [12]. Generally speaking, this type of obfuscated traffic has artificially high entropy, which means the injected noise and entropy itself become a distinguishing feature.

Protocol mimicry The next logical iteration for disguising protocol behavior was to mimic some uncensored protocol. Instead of simply removing features, this technique shapes covert traffic features to match those of another innocuous protocol in order to make it more difficult to censor covert traffic. Examples of this technique in the literature include SkypeMorph and StegoTorus [13, 14]. SkypeMorph learns the distribution of packet size and timing for Skype voice and conforms the covert traffic packets to fit this distribution. StegoTorus similarly aims to mimic innocuous traffic, namely HTTP or VoIP protocols like Skype and Ventrilo, by applying steganography. Both SkypeMorph and StegoTorus were designed to disguise Tor traffic. An example of a standalone protocol mimicry system is CensorSpoofer, which aims to mimic VoIP traffic to hide covert web traffic [15]. Another approach to protocol mimicry is called Format Transforming Encryption. This approach specifies a regular expression to represent the target protocol's packet properties, and format transforming encryption produces Tor packets that match the regular expressions [16, 17].

Houmansadr et al. demonstrated a key observation that generally defeated all attempts at protocol mimicry [18]. They show that a protocol mimicry approach must perfectly reimplement all features of the target protocol, not just the statistical or syntactic properties of its traffic. Taking Skype and SkypeMorph as an example, the authors could trivially detect SkypeMorph due to observations like missing control channels, lack of communication to login servers, reaction to blocked default ports, etc. Furthermore, they showed that even a correct implementation of the complex distributed system could potentially be identified, because it *didn't* incorporate bugs that were found in the real Skype protocol implementation.

Protocol tunneling. Various protocols have been used as the tunnel for covert communication [19–26] and we focus on multimedia protocol tunnels in this paper. In a multimedia protocol tunnel, the censored data is first encoded into the media normally communicated by the target audio/video protocol (e.g., modulated into an audio signal) and then transmitted through a running instance of the target protocol. The first instantiation of this idea is FreeWave [27]. The FreeWave protocol tunnel designs a modem-based proxy that could communicate IP traffic through an audio channel. That audio channel is carried by a Skype VoIP call between the client in a censored network and a server in the uncensored network. By using the actual Skype application to carry the data, FreeWave guarantees that the semantics of the protocol is truly indistinguishable from a normal Skype phone call. This effectively eliminates the problems with protocol mimicry, but reduces throughput and increases latency. Video-based multimedia protocol tunnels, which encode covert data into video streams, include Facet, CovertCast, DeltaShaper, Protozoa, and Stegozoa [2–4, 22, 28].

Geddes et al. demonstrated how protocol tunnels are subject to content mismatch attacks [5]. Nearly all multimedia communication protocols use variable bitrate encoding to reduce network usage by adapting their data transmission rate based on the properties of the media content. For example, the OPUS codec (used by many VoIP applications including Skype) can adjust its bitrate between 6 kbps and 510 kbps depending on factors including network conditions and audio quality and drastically reduces bitrate during silence or background noise [29]. This effectively leaks information related to the content carried by the variable-bitrate-encoded and encrypted voice channel [30]. This observation can be exploited to distinguish true Skype conversations from FreeWave transmissions, for example.

This type of content mismatch attack has been identified both for audio-based protocol tunnels and video-based proto-

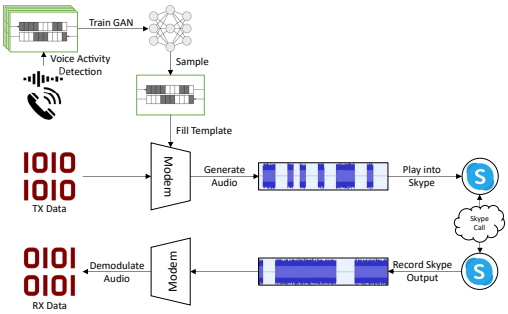

Figure 2: This is the complete Voiceover architecture, including model training, data transmission, and data reception. The topmost portion (Voice Activity Detection and Train GAN) presumably will happen only once or very rarely. The model produced by that training can be used repeatedly to generate sample conversation templates. The remainder of the diagram will be repeated for every transmission (e.g., for every VoIP call to transmit and receive some data).

col tunnels, where variable bitrate encoding is leaking information about what content is being tunneled. When combined with machine learning classification techniques, this exploit allows a censor to identify new tunnels without needing to manually determine the distinguishing properties [31]. Our work on Voiceover attempts to mitigate content mismatch attacks in multimedia protocol tunnels.

3 Voiceover Design

In this section, we present the design of Voiceover, an audio-based protocol tunnel that uses generative models to mitigate content mismatch attacks. The architecture of Voiceover is shown in Figure 2. Our key insight follows naturally from observations about content mismatch attacks:

Content mismatch attacks. Content mismatch attacks exploit the fact that differences between two classes of media content cause differences in the network traffic that carries that content. The attack works because there exists some distribution of properties for the "real" class of content (e.g., an actual human conversation). Content carried by protocol tunnels does not have properties that fit within that distribution. Therefore, the network traffic produced by a protocol tunnel does not fit within the distribution of network traffic produced by normal use of the protocol. Our goal is to encode our covert data into media content that possesses properties within that existing "real content" distribution; the natural tool for this is generative machine learning models.

Learning the distribution. The most immediately apparent form of data leakage by variable bitrate voice codecs is often the difference between speaking and not speaking. This is due to the widespread usage of the OPUS codec, which uses a "discontinuous transmission" mode to send only one frame

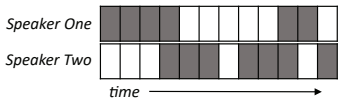

Figure 3: Conversation timing is encoded as a two-dimensional array. Each row is a speaker, and each column is a unit of time. The value in each cell encodes either speech or silence as either positive or negative one.

every 400 milliseconds during silence or background noise [29]. For our initial design, we target this specific leakage by trying to learn the cadence and timing of speakers in real two-person conversations.

Generative adversarial networks (GANs) have shown great results in learning "realistic" image distributions and generating new random samples which are statistically and visually similar to the training images [32]. Our learning model will be a reduction of this same problem, where we structure our data in a format similar to an image. To learn the cadence and timing of a conversation, we will operate on *conversation templates*, visualized in Figure 3. We define a conversation template as a two-dimensional array: each row represents one speaker and each column represents one unit of time. In each cell of this template, the value will be $+1$ or -1 to represent silence or speech. In this format, temporal locality in the conversation is translated into spatial locality in the template. We use a deep convolutional generative adversarial network (DCGAN) and Figure 4 visualizes the final DCGAN architecture for our model. The DCGAN architecture was tuned by experiments on the training data.

We use a mix of linguistics data collected in various studies over the years on telephone conversations for training. These datasets consist of two-person conversations primarily in English with varying lengths between five minutes and thirty minutes. The datasets are listed in Table 1. In order to transform these audio recordings into our learnable format, we use a state-of-the-art voice activity detection (VAD) implementation to determine for each time period whether or not each speaker is speaking. We modified a sliding window voice activity detection algorithm based on the VAD module implemented for WebRTC [33, 34]. Since the GAN requires a constant-sized format, we split the conversations into 5-minute samples at a resolution of 1-second time periods. These parameters are tunable for future experiments to determine the effects of both lower-order and higher-order timing properties. The result of this process is a collection of samples in the format described above with dimensions of 2 by 300. The impact of the quantity and quality of this data is discussed in section 5.

Encoding and modulating the data. Our protocol tunnel must deliver binary data through an audio channel and persist through errors caused by noise, compression, and transcoding. We view this challenge as a classic data transmission problem

Data	Desc.
CALLHOME Speech Corpus [35]	120 * 30 min
Linguistic Data Consortium Switchboard [36]	2400 * <15 min
2011 NIST Language Recognition [37]	400 * 5/10 min

Table 1: The original linguistic dataset used to train our model contains approximately 400 hours of two-person conversations.

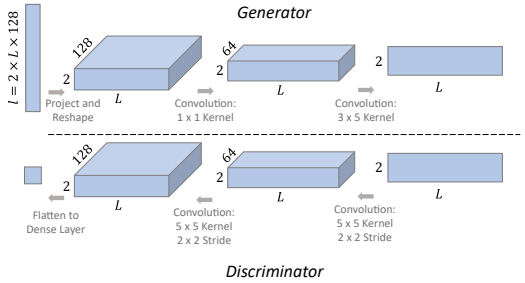

Figure 4: The generator (top) and discriminator (bottom) for learning the data distribution of conversation timing is derived from existing DCGAN architectures from the image domain.

in signal processing. Naturally, the solution that follows is a coding and modulation scheme designed around the noisy channel. Similarly to how radio frequency transmissions are modulated to establish wireless network connections, we will use quadrature amplitude modulation (QAM). Specifically, we will follow a similar configuration to that used in the FreeWave paper [27]. In this configuration, we combine two common signal-processing techniques: convolutional coding and phase shift keying. Voiceover applies convolutional coding to a given plaintext before applying QAM to modulate the encoded data into an audio signal.

Handling application-layer transformations. Since Voiceover is an audio-based multimedia protocol channel, the integrity of the message data is dependent on the integrity of the audio signal transmitted over the protocol channel. However, many voice codecs will distort audio signals that are not similar to human speech [38]. Since Voiceover transmits a QAM-modulated audio signal, it is important for the sake of the reliability of the censorship circumventing protocol to be able to recover from application-layer distortions and transformations of the transmitted audio signal.

To recover from these application-layer transformations, we take an approach inspired by notions of framing and encapsulation in networking. To transmit a plaintext message, we first break up the plaintext into smaller substrings and add headers to these substrings to create a 'message frame.' We then modulate each of these message frames into audio signals, which we treat as 'audio frames.' Rather than sending one continuous audio signal containing the entire plaintext, we send an audio signal composed of these smaller audio frames which are modulated from smaller portions of the plaintext. To verify the integrity of the contents of the message frame, we include a CRC32 checksum of the encoded message frame as part of the message frame header which can be verified by the receiver during decoding. If a message frame integrity check has failed, Voiceover determines that an application-layer transformation of the audio signal has occurred and ignores that message frame when reconstructing the full plaintext. Besides, Voiceover sends each frame k times (where k is the redundancy factor) in case of packet loss. To facilitate potential user-driven retransmission of these message frames, Voiceover includes a two-byte frame number as part of the header for each encoded message frame. In total, there are 6 bytes of overhead per message frame: 4 bytes for a CRC32 checksum and 2 bytes for a message frame number.

Applying the model. Voiceover can create realistic timing templates for a voice conversation and transmit data as an audio waveform. Next, we discuss how to combine these components. We focus on the scenario where both ends of the channel have either a pre-shared template or the same generative model and a pre-shared seed for producing the template. Each endpoint must also choose a unique row of the template. This bootstrapping process will need to be designed based on specific use cases and corresponding security assumptions. At this point, each endpoint will behave symmetrically.

Scanning over the single row of the template, Voiceover will determine the length of each uninterrupted "speaking" period. Based on the parameters of the encoding and modulation, Voiceover computes the number of bytes to send which will produce exactly that duration of audio. During "silent" periods in the template, Voiceover simply does not transmit any audio. During "overlapping" periods in which both endpoints are 'speaking,' Voiceover will modulate null bytes which will be discarded during demodulation. This was done to improve goodput as audio interference could cause application-layer transformations of the transmitted audio signal. Concurrently, Voiceover will record all audio received from the VoIP channel and process it with the receiving side of the demodulation and decoding process. This forms the base audio protocol tunnel through which we can transmit and receive binary data.

4 Prototype Implementation and Evaluation

Voiceover implementation. We implemented the GAN component in Python using the `Keras` framework based on an open source reference implementation [39, 40]. The encoding and modulation logic was constructed in Python using a signal processing library based on several samples of QAM designs [41, 42]. The protocol tunnel was chosen to be Skype for Web. We automate many aspects of the audio transmission process for Voiceover using Linux pulseaudio and virtual audio cable, and Selenium. To improve the performance of the Voiceover demodulation process, we divide the received

audio signal into its constituent audio signal frames, and parallelize the demodulation of the audio signal frames among all processor cores present on a Voiceover receiver endpoint. This change improves demodulation runtime by a constant factor. There are five audio modulation parameters that we expose to the user to allow for finer tuning of a modulated audio signal: sampling frequency, carrier frequency, baud rate (or equivalently, symbol period), message frame length, and message frame redundancy factor. See Appendix B for more details. The full implementation source is made available on GitHub.

Strawman baseline implementation. Without access to any implementations of prior work on audio-based protocol tunnels (which are not publicly available), we must implement our own baseline to represent the existing techniques. For this evaluation, we simply need this strawman implementation to be an audio-based protocol tunnel that does not attempt to shape the audio it transmits. As such, we can create the strawman implementation using the same Voiceover architecture but with a template that indicates continuous speech in both directions. Each side of the tunnel will be continuously transmitting as if exchanging covert data.

Data collection Our testing platform consists of two Skype endpoints, each running in its own virtual machine on a single host. The virtual machines are connected to an isolated virtual network with access to the internet through a NAT. One of the virtual machines also runs tcpdump during transmissions to collect the raw packet traces. For this data collection, the operating system chosen for the virtual machines was Ubuntu 18.04. Two endpoints are coordinated to use Skype for Web to play the audio.

The real conversation data is based on audio recordings from our original training data in Table 1. We then apply a randomly GAN-generated template to the real data to create Voiceover audio. For strawman, we use a template allowing constant simultaneous transmission as described above to generate the audio file. We collect the data by splitting each conversation into isolated speaker channels and playing one channel on each virtual machine. In total, we collected 4 hours of each real, strawman, and Voiceover transmission.

4.1 Modeling the Adversary

Next, we evaluate the efficiency of Voiceover against a real-time ML-based network traffic classifier. The classifier plays the role of a memory-limited adversary whose goal is to identify protocol tunnels in real-time through machine-learning classification of network flows using packet-level features. Here, we represent the memory limitation of the adversary in the number of consecutive packets that it will be able to consider during network flow classification. We limit this to 1000 packets per flow, but we also evaluate Voiceover when this limit is decreased to 500 packets.

The classifier leverages various information commonly used in traffic analysis [43, 44]: time, packet size, and packet direction (+/-1). We extend the nPrint [45] framework to automatically extract features from a packet. Each packet is represented with 18 features and each set of 1000 packets is thus represented as a sample containing 18,000 features. This feature representation similarly applies to sets of 500 packets. We automated the process of feature selection, model selection, and hyperparameter tuning using automated machine learning (AutoML). We used the AutoGluon-Tabular implementation which is bundled with the nPrint framework [45, 46]. The models chosen are from the AutoGluon implementation, which includes neural networks, random forests, and CatBoost trees, to name a few [46]. We model the adversary as any machine learning classifier trained by AutoGluon-Tabular. We use AutoGluon-Tabular to train machine learning models on two binary classification cases: real vs. strawman and real vs. Voiceover.

4.2 Evaluation Results

We compare the performance of the AutoGluon-Tabular models trained on the real vs. strawman case to that of the AutoGluon-Tabular models trained on the real vs. Voiceover case for the 1000 packet setting by comparing their precision-recall (PR) curves and receiver operating characteristic (ROC) curves. Given the processed packet capture data described above, the AutoGluon-Tabular models for the real vs. strawman case were trained on 2,160 samples, validated on 500 samples, and tested on 1,140 samples; the AutoGluon-Tabular models for the real vs. Voiceover case were trained on 2,197 samples, validated on 500 samples, and tested on 1,157 samples. The evaluation metric for the AutoGluon-Tabular training process for both real vs. strawman and real vs. Voiceover cases was chosen to be accuracy. In the remainder of this section, we refer to an adversary trained to identify strawman transmissions as meaning an adversary using the machine learning models trained by AutoGluon-Tabular on the real vs. strawman case. We use similar language meant to describe an adversary trained to identify Voiceover transmissions.

An adversary using a classifier to identify the strawman transmissions could do so with far greater certainty than using a classifier to identify Voiceover transmissions. Figure 5 compares the PR curves of the best-performing model trained by AutoGluon-Tabular for the real vs. strawman case, referred to as the strawman classifier, to that of the best-performing model trained by AutoGluon-Tabular for the real vs. Voiceover case, referred to as the Voiceover classifier. We observe that the adversary is capable of classifying strawman transmissions with extremely high precision and recall. However, the adversary does not achieve the same performance in classifying Voiceover transmissions, as **the Voiceover classifier incurs a significant decrease in precision even when the recall rate is at 0.1.** Now comparing the ROC curves of

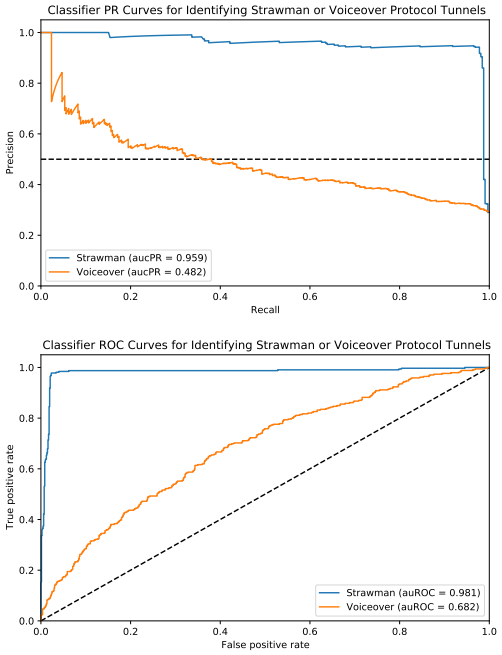

Figure 5: The Precision-Recall (Upper) and ROC curves (Lower) of the best-performing attacks against Strawman and Voiceover. The dashed line represents random guessing.

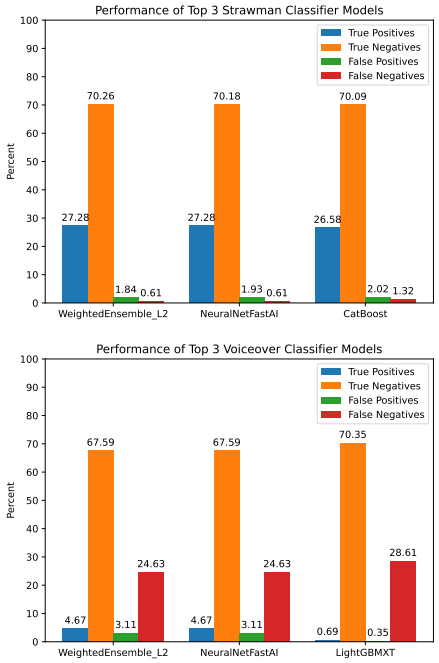

Figure 6: The top three models against Strawman (Upper) and Voiceover (Lower).

# of Packets	500	1000
Strawman aucPR	0.93	0.96
Voiceover aucPR	0.50	0.48
Strawman auROC	0.97	0.98
Voiceover auROC	0.68	0.68

Table 2: The efficiency of the best-performing attacks when using different numbers of packets for attacks.

the best performing models of the strawman and Voiceover classifiers in Figure 5, we observe that with an extremely small false positive rate, the adversary could identify nearly all of the strawman transmissions. On the other hand, for the same false positive rate, the adversary could identify less than 20% of Voiceover transmissions and **the Voiceover classifier must incur more false positives to achieve a higher true positive rate.** This discrepancy is also evident in the auROC for each classifier, where the auROC for identifying Voiceover is less than 0.7 while the auROC for identifying the strawman transmissions is nearly 1.0.

The bar charts given in Figure 6 represent the raw confusion matrix values of the top three performing models that AutoGluon-Tabular evaluated for the real vs. strawman and real vs. Voiceover cases each as percentages of all tested samples. We can see that **for the top three models trained to identify strawman transmissions, the top three models trained to identify Voiceover transmissions all incurred a large increase in false negatives**, indicating that the models had more difficulty in being able to distinguish Voiceover transmissions from real transmissions.

Lastly, we evaluate the performance of Voiceover when the number of consecutive packets that the adversary is allowed to consider for flow classification is varied. We evaluated the performance of Voiceover when this limit was decreased to 500 packets. This changes the classification problem but varies the amount of information that the adversary is allowed to access in each packet flow sample.

For the 500 packet setting, the AutoGluon-Tabular models for the real vs. strawman case were trained on 4,822 samples, validated on 536 samples, and tested on 2,297 samples. The AutoGluon-Tabular models for the real vs. Voiceover case were trained on 4,888 samples, validated on 544 samples, and tested on 2,329 samples. The evaluation metric for the AutoGluon-Tabular training process for both real vs. strawman and real vs. Voiceover cases in the 500 packet setting was again accuracy. Table 2 gives the area under ROC curves of the best-performing model trained by AutoGluon-Tabular for the real vs. strawman case compared to that of the model trained by AutoGluon-Tabular for the real vs. Voiceover case when the packet limit is varied. The results show that **even when the number of packets the adversary is allowed to consider is varied, Voiceover reduced classification accuracy**.

These results support both of our hypotheses and illustrate the value of the audio-shaping component of Voiceover in decreasing classification accuracy for machine-learning

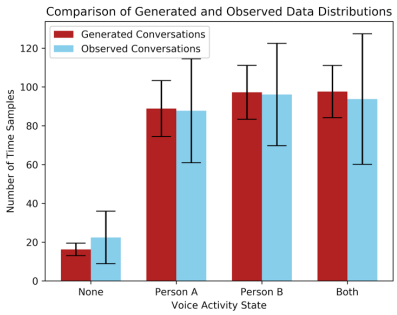

Figure 7: We compare the histogram distribution of speaking states between generated samples and truly observed training samples. Here the right/blue bars represent the count of time periods for each speaking state (Neither Speaker, Speaker A, Speaker B, and Both Speakers).

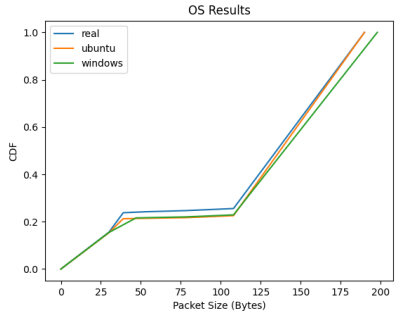

Figure 8: The packet size CDFs of Voiceover transmissions and real conversations across different operating systems is shown.

classifiers. First, the machine learning classifiers trained by AutoGluon-Tabular were able to identify a substantial portion the strawman transmissions with minimal consequences in falsely identifying the real conversations. Second, applying Voiceover's GAN to shape audio transmissions made the performance of the AutoGluon-Tabular classifiers significantly worse.

4.3 GAN Output Distribution

We evaluate the output distribution of Voiceover's GAN output timing templates. Training the GAN model over 10,000 tandem epochs using an Adam optimizer on the discriminator's binary cross-entropy loss, we create a model that produces realistic templates fitting within the distribution of the real training data templates. As a high-level test of this model, we study the distribution of speaking states between generated samples and training samples. The four possible states of speaking in a two-person conversation are None, Person A, Person B, and Both. Figure 7 shows a comparison of the distributions. Here we can see that the GAN model is generating samples that reasonably fit within the distribution of real training samples.

4.4 Packet Size Analysis

While packet size was leveraged as a feature in our machine learning evaluation component, we also found that **Skype for Web is a protocol tunnel that is immune to packet size analysis**. We collected packet traces from only one Voiceover endpoint that is transmitting an outgoing signal. We then calculated the cumulative distribution function (CDF) of packet sizes associated with Skype network traffic using packet traces collected from outgoing Voiceover audio transmissions and compared it to the CDF of packet sizes associated with Skype network traffic using packet traces collected from outgoing real audio transmissions. Qualitatively, unobservability is achieved when the CDF of packet size from the real audio transmission is practically identical to that of the Voiceover audio transmission. We compared packet size CDFs across different operating systems to further show that these results hold even on different OS platforms. We further elaborate on our experimental design choices and compared packet size CDFs across different Voiceover audio modulation parameters in Appendix C.

In our operating systems comparison, the operating systems were chosen to be Ubuntu 18.04 and Windows 10. We fixed Voiceover modulation parameters to their default values and fixed the timing template used to shape the covert audio signal. The covert plaintext messages to be transmitted by Voiceover were randomly generated. We collected over 2 hours worth of packet traces for each operating system and over 4 hours worth of packet traces of real audio transmissions. The CDF of packet sizes for each type of audio transmission in this experiment is shown in Figure 8.

We find that across different operating systems and different Voiceover modulation parameters (the latter results are given in Appendix C), the CDFs of packet sizes are nearly identical to each other which suggests that **Skype for Web provides for a protocol tunnel resistant to packet size analysis.** Moreover, since the packet size CDFs are identical across different operating systems, this bolsters the usability of Voiceover across popular OS platforms.

4.5 Throughput Analysis

We evaluate the overall usability of Voiceover as a communication tool by analyzing its goodput. Since Voiceover breaks up plaintext messages into message frames for modulation, we focus on the throughput of solely the plaintext message (i.e. goodput) since each message frame possesses overhead. Voiceover transmits shaped audio signals that have a maximum duration of five minutes. Assuming that no audio shaping is applied, that is, a strawman transmission, the good-

Parameter	Goodput (bytes/s)
Skype Theoretical Limit	750 - 63,750
Strawman w/ Defaults	62.32
Strawman w/ No Redundancy	124.68
Strawman w/ Redundancy Factor = 3	41.56
Strawman w/ Frame Length = 3	36.92
Strawman w/ Frame Length = 12	84.96
Voiceover w/ Defaults	31.16
Voiceover w/ No Redundancy	62.32
Voiceover w/ Redundancy Factor = 3	20.78
Voiceover w/ Frame Length = 3	18.46
Voiceover w/ Frame Length = 12	42.48

Table 3: Goodput in bytes under different parameters.

put of Voiceover with default audio modulation parameters and default reliability layer parameters will be 62.32 bytes per second. However, Voiceover will shape audio signals by adding periods of silence when a message is being modulated into an audio signal. Thus, the goodput of Voiceover will be strictly less than 62.32 bytes per second. The actual goodput of Voiceover will be dependent on the timing template used, since the timing template determines when Voiceover will transmit the signal and when it will transmit silence.

There are certain tradeoffs and optimizations that could increase goodput. For example, one could forego the application of redundancy in the reliability layer to increase throughput at the expense of message recovery. Since the default setting is to add one redundant frame, removing this redundancy would lead to a doubling of goodput. Conversely, adding more redundant frames would decrease goodput. Another tradeoff is the size of each frame. If one were to choose larger frame sizes, goodput would increase as their per-frame overhead would decrease. On the other hand, if one were to choose smaller frame sizes to improve message recoverability, goodput would decrease as more frames would be required to modulate an entire signal, leading to an increase in per-frame overhead.

It is worth mentioning that since Voiceover uses Skype as an audio protocol tunnel, the throughput of Voiceover will have a theoretical limit equal to that of the Skype audio protocol. For the Skype audio codec, the limit is between 6 and 510 kilobits per second, or equivalently, between 750 and 63,750 bytes per second [29]. These throughput results suggest that Voiceover is more suitable for less-demanding applications in terms of data throughput. A summary of this throughput analysis showing different trade-offs is given in Table 3.

5 Limitations and Future Work

Use realistic training data. Our preliminary analysis uses telephone conversations captured from several separate linguistic studies to approximate VoIP conversations. Our future work considers collecting a large number of real VoIP conversations for evaluation.

Consider advanced adversaries. Our work mostly focuses on content mismatch caused by timing, while there are more advanced content mismatch attacks (e.g., inferring the phrases or words that were spoken [47]). We will consider advanced attacks and perform a more rigorous security analysis. Additionally, we will examine the trade-off between resource usage and attack efficiency to justify the assumption of memory-limited adversaries.

Improve throughput. We plan to further reduce performance overhead to improve the throughput of Voiceover, by tuning the parameters discussed in Appendix B. Note that every multimedia protocol tunnel is strictly limited to a throughput below the throughput of the target protocol so Voiceover is not suitable for high-throughput applications. That said, Voiceover is useful, e.g., as an out-of-band channel for sharing secret keys for censorship circumvention tools.

6 Conclusion

This paper presents Voiceover, a censorship circumventing protocol tunnel that mitigates content mismatch attacks using generative modeling. Our technique acknowledges that a mismatch of media properties in a protocol tunnel will cause distinguishing features in the network traffic produced by that tunnel. We address this by constructing a generative machine-learning model that learns the distribution of timing properties from real two-person conversations. This generative model acts as a template to conform our protocol tunnel transmissions to fit within a distribution of realistic conversations. Voiceover, therefore, resists detection not only by analysis of the protocol implementation but also resists detection by censors employing more advanced traffic analysis techniques that leverage machine learning. We provide open-source access to our prototype implementation and contribute part of our signal processing implementation to a widely used research library. With this prototype, we show that a protocol tunnel without generative modeling can be reliably identified by machine-learning classifiers and that when Voiceover introduced conversation modeling based on timing properties, classifier performance decreased.

7 Acknowledgments

We would like to thank Vinod Yegneswaran and the anonymous reviewers for their constructive feedback. This work was supported in part by the National Science Foundation under grants CNS-1553437, CNS-1704105, CNS-2131938, the Open Technology Fund, and by the United States Air Force and DARPA under Contract No. FA8750-19-C-0079. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force, DARPA, or any other sponsoring agency.

References

[1] Z. Wang, Y. Cao, Z. Qian, C. Song, and S. V. Krishnamurthy, "Your state is not mine: a closer look at evading stateful internet censorship," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 114–127.

[2] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 163–172.

[3] D. Barradas, N. Santos, and L. Rodrigues, "Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 5–22, 2017.

[4] R. McPherson, A. Houmansadr, and V. Shmatikov, "Covertcast: Using live streaming to evade internet censorship," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, pp. 212–225, 2016.

[5] J. Geddes, M. Schuchard, and N. Hopper, "Cover your acks: Pitfalls of covert channel censorship circumvention," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 361–372.

[6] B. Wiley, "Dust: A blocking-resistant internet transport protocol," *Technical rep ort. http://blanu. net/Dust. pdf*, 2011.

[7] P. Winter, T. Pulls, and J. Fuss, "Scramblesuit: A polymorphic network protocol to circumvent censorship," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 213–224.

[8] "Tor project, obfsproxy2." [Online]. Available: https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs2/obfs2-protocol-spec.txt

[9] "Tor project, obfsproxy3." [Online]. Available: https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt

[10] "Tor project, obfsproxy4." [Online]. Available: https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt

[11] "Racecar project, obfsproxy5." [Online]. Available: https://racecar.cs.georgetown.edu/software/

[12] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton, "Seeing through network-protocol obfuscation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 57–69.

[13] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 97–108.

[14] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "Stegotorus: a camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 109–120.

[15] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, "Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 121–132.

[16] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Protocol misidentification made easy with format-transforming encryption," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 61–72.

[17] K. P. Dyer, S. E. Coull, and T. Shrimpton, "Marionette: A programmable network traffic obfuscation system," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 367–382.

[18] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 65–79.

[19] "Tor project, meek." [Online]. Available: https://gitlab.torproject.org/legacy/trac/-/wikis/doc/meek

[20] "Tor project, snowflake." [Online]. Available: https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transports/snowflake/-/wikis/home

[21] D. Fifield, "Turbo tunnel, a good way to design censorship circumvention protocols," in *10th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 20)*, 2020.

[22] D. Barradas, N. Santos, L. Rodrigues, and V. Nunes, "Poking a hole in the wall: Efficient censorship-resistant internet communications by parasitizing on webrtc," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 35–48.

[23] W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov, "Sweet: Serving the web by exploiting email tunnels," *arXiv preprint arXiv:1211.3191*, vol. 13, 2012.

[24] C. Brubaker, A. Houmansadr, and V. Shmatikov, "Cloudtransport: Using cloud storage for censorship-resistant networking," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2014, pp. 1–20.

[25] B. Hahn, R. Nithyanand, P. Gill, and R. Johnson, "Games without frontiers: Investigating video games as a covert channel," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 63–77.

[26] P. Vines and T. Kohno, "Rook: Using video games as a low-bandwidth censorship resistant communication platform," in *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, 2015, pp. 75–84.

[27] A. Houmansadr, T. J. Riedl, N. Borisov, and A. C. Singer, "I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention." in *NDSS*, 2013.

[28] G. Figueira, D. Barradas, and N. Santos, "Stegozoa: Enhancing webrtc covert channels with video steganography for internet censorship circumvention," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 1154–1167.

[29] J. M. Valin, K. Vos, and T. Terriberry, "Definition of the opus audio codec," Internet Requests for Comments, RFC Editor, RFC 6716, 9 2012. [Online]. Available: https://tools.ietf.org/html/rfc6716

[30] C. Perkins and J. M. Valin, "Guidelines for the use of variable bit rate audio with secure rtp," Internet Requests for Comments, RFC Editor, RFC 6562, 3 2012. [Online]. Available: https://tools.ietf.org/html/rfc6562

[31] D. Barradas, N. Santos, and L. Rodrigues, "Effective detection of multimedia protocol tunneling using machine learning," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 169–185.

[32] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[33] Google, "Webrtc," https://webrtc.org/.

[34] wiseman, "py-webrtcvad," https://github.com/wiseman/py-webrtcvad/releases/tag/2.0.10.

[35] A. Canavan, D. Graff, and G. Zipperlen, "Callhome american english speech," https://catalog.ldc.upenn.edu/LDC97S42.

[36] J. J. Godfrey and E. Holliman, "Switchboard-1 release 2," https://catalog.ldc.upenn.edu/LDC97S62.

[37] C. Greenberg, A. Martin, D. Graff, K. Walker, K. Jones, and S. Strassel, "2011 nist language recognition evaluation test set," https://catalog.ldc.upenn.edu/LDC2018S06.

[38] A. Dhananjay, A. Sharma, M. Paik, J. Chen, T. K. Kuppusamy, J. Li, and L. Subramanian, "Hermes: data transmission over unknown voice channels," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, 2010, pp. 113–124.

[39] eriklindernoren, "Keras-GAN," https://github.com/eriklindernoren/Keras-GAN.

[40] TensorFlow Core Tutorials, "Deep convolutional generative adversarial network." [Online]. Available: https://www.tensorflow.org/tutorials/generative/dcgan

[41] V. Taranalli, "CommPy: Digital Communication with Python," https://github.com/veeresht/CommPy.

[42] "Baseband signal upconversion and iq modulation and demodulation." [Online]. Available: https://dspillustrations.com/pages/posts/misc/baseband-up-and-downconversion-and-iq-modulation.html

[43] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1928–1943.

[44] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, "Tik-tok: The utility of packet timing in website fingerprinting attacks," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 3, 2020.

[45] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," 2021.

[46] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," *arXiv preprint arXiv:2003.06505*, 2020.

[47] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose, "Phonotactic reconstruction of encrypted VoIP conversations: Hookt on fon-iks," in *2011 IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 3–18.

A Per-Model Evaluation of Voiceover

We present the PR and ROC curves for all machine learning classifiers trained by AutoGluon-Tabular for the real vs. strawman and real vs. Voiceover classification cases in the

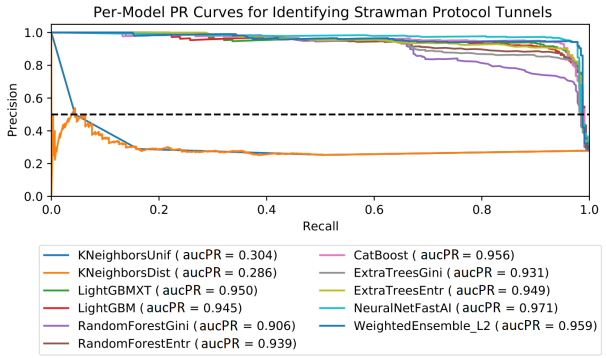

Figure 9: These PR curves represent the possible tradeoffs between recall and precision when identifying the strawman transmission or the Voiceover transmission. The dashed line represents a precision rate of 50% while a perfect classifier would be a horizontal line with a precision of 1. These results indicate a substantial decrease in the performance of the adversary when Voiceover introduces the conversation modeling on top of the strawman technique.

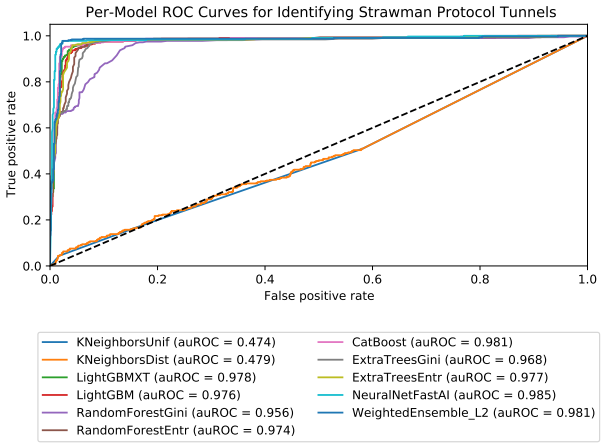

Figure 10: These ROC curves represent the possible tradeoffs between false positives and true positives when identifying the strawman transmission or the Voiceover transmission. The dashed line represents random guessing while a perfect classifier would be a horizontal line at 100% true positive rate. These results indicate a substantial decrease in the performance of the adversary when Voiceover introduces the conversation modeling on top of the strawman technique.

1000 packet setting. The results are given in Figure 9, Figure 10, Figure 11, and Figure 12. Note that AutoML did not train an XGBoost model in the real vs. strawman classification case. However, we do not expect the XGBoost model to perform significantly worse than the other models trained in the real vs. strawman classification case, as XGBoost has previously been shown to have been effective in classifying other multimedia-based protocol tunnels [31].

The results show that for the real vs. Voiceover classification case, each model trained by AutoGluon-Tabular experienced a degradation in performance compared to its corresponding model trained by AutoGluon-Tabular for the real vs. strawman classification case. This is attributable to Voiceover's GAN component that is applying audio shaping based on timing properties.

B Voiceover Parameters

There are five audio modulation parameters that we expose to the user to allow for finer tuning of a modulated audio signal: sampling frequency, carrier frequency, baud rate (or equivalently, symbol period), message frame length, and message frame redundancy factor. The default sampling frequency, carrier frequency, and symbol period were chosen to be 8 kHz, 2 kHz and 0.625 ms, respectively. A carrier frequency of 2 kHz ensures that the lowest sample rate of the OPUS codec will still fully capture the signal (see the Nyquist-Shannon sampling theorem). The 0.625ms symbol period aims to maximize throughput in transmission over the VoIP channel. The message frame length is the length in characters of a plaintext substring that will be modulated into an audio frame as described in section 3. Note that message frame length refers to the message payload, not the full length of the message frame in bytes. This value was set to 6, which is roughly the average length of an English word plus a space character. The message frame redundancy factor determines the number of modulated audio frames that will be sent after a plaintext substring has been modulated into an audio frame as described in the previous subsection. This value was set to 2, meaning the original audio frame as well as a copy of that frame will be sent as part of the full audio signal for transmission.

C Voiceover Parameter Packet Size Analysis

We present packet size CDF graphs for Voiceover transmissions while varying Voiceover audio modulation parameters. In these experiments, we only consider outgoing Voiceover transmissions, that is, a Voiceover transmission in which only one endpoint is transmitting. This allows us to isolate the network traffic associated with a Voiceover endpoint for the purposes of analyzing packet sizes. All of these experiments were conducted using Skype for Web on Ubuntu 18.04. For these experiments, we relaxed the requirement that each Voiceover transmission be 5 minutes long (the full length of a timing template sampled from Voiceover's GAN) and instead have each transmission be 30 seconds long. This was done in order to collect sufficiently large amounts of packet trace data on

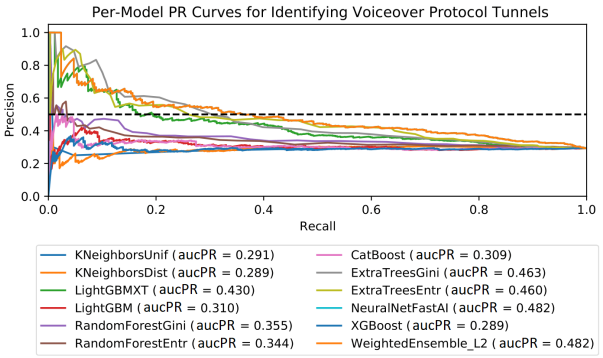

Figure 11: These PR curves represent the possible tradeoffs between recall and precision when identifying the strawman transmission or the Voiceover transmission. The dashed line represents a precision rate of 50% while a perfect classifier would be a horizontal line with a precision of 1. These results indicate a substantial decrease in the performance of the adversary when Voiceover introduces the conversation modeling on top of the strawman technique.

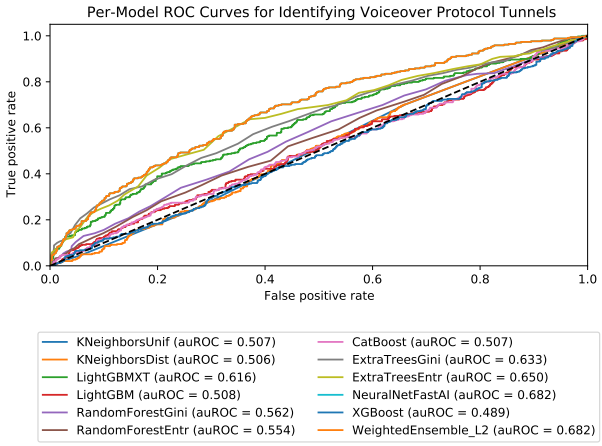

Figure 12: These ROC curves represent the possible tradeoffs between false positives and true positives when identifying the strawman transmission or the Voiceover transmission. The dashed line represents random guessing while a perfect classifier would be a horizontal line at 100% true positive rate. These results indicate a substantial decrease in the performance of the adversary when Voiceover introduces the conversation modeling on top of the strawman technique.

unique Voiceover transmissions. This does not impact the reliability of our results, as we are only concerned with obtaining enough packet size data from Skype traffic. We calculated packet size CDFs varying four different Voiceover modulation parameters: carrier frequency, sampling frequency, baud rate, and message frame length. These parameters were chosen for evaluation as changing these parameters results in changes to a Voiceover-modulated audio signal which could create opportunities for content mismatch.

We also calculated packet size CDFs for bidirectional Voiceover communication, that is, collecting packet trace data in the situation where two Voiceover endpoints are communicating with each other, rather than only looking at the outgoing traffic for one Voiceover endpoint. This experiment aims to evaluate Voiceover in a simulated real-world setting.

For all graphs, we also give the CDF of packet sizes from a real audio transmission as a comparison to demonstrate unobservability. For all experiments, 4 hours worth of real audio transmission packet trace data was collected.

C.1 Carrier Frequency

In our carrier frequency comparison, we compared carrier frequencies of 1, 2, and 3 KHz. We fixed all other Voiceover modulation parameters to their default values and fixed the timing template used to shape the audio signal. The covert plaintext messages to be transmitted by Voiceover were randomly generated. We collected roughly 1 hours worth of packet traces for each carrier frequency variable. The CDF of packet sizes for this experiment is shown in Figure 13.

C.2 Sampling Frequency

In our sampling frequency comparison, we compared sampling frequencies of 8 and 16 KHz. We fixed all other Voiceover modulation parameters to their default values and fixed the timing template used to shape the audio signal. The covert plaintext messages to be transmitted by Voiceover were randomly generated. We collected roughly 1 hours worth of packet traces for each sampling frequency variable. The CDF of packet sizes for this experiment is shown in Figure 14.

C.3 Baud Rate

In our baud rate comparison, we compared baud rates of 750 and 1500 symbols per second. We fixed all other Voiceover modulation parameters to their default values and fixed the timing template used to shape the audio signal. The covert plaintext messages to be transmitted by Voiceover were randomly generated. Since the baud rate affects the amount of a plaintext message that can be modulated in a given second, the message lengths of the random plaintexts cannot be kept constant across the baud rate variables (each Voiceover transmission must be roughly 30 seconds long by our experimental design). However, we expect that differing message lengths will have no effect on the packet size distribution. Further note that we did not compare the default baud rate of 1599 symbols per second. Baud rates of 750 and 1500 symbols per second were chosen to make setting plaintext message

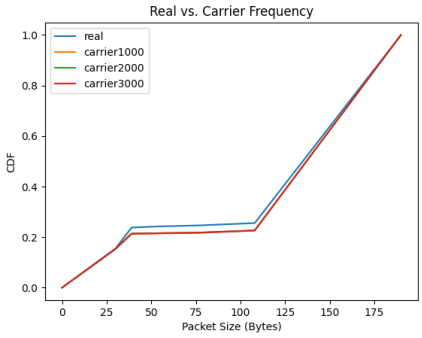

Figure 13: The packet size CDFs of Voiceover transmissions across different carrier frequencies is shown. The CDF of real conversations is also shown as a comparison. We can see that the packet size CDFs for all three carrier frequencies closely matches that of the packet size CDF of a real conversation.

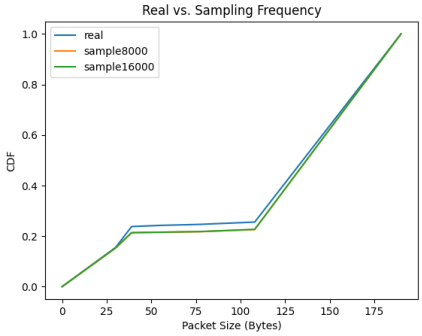

Figure 14: The packet size CDFs of Voiceover transmissions across different sampling frequencies is shown. The CDF of real conversations is also shown as a comparison. We can see that the packet size CDFs for both sampling frequencies closely matches that of the packet size CDF of a real conversation.

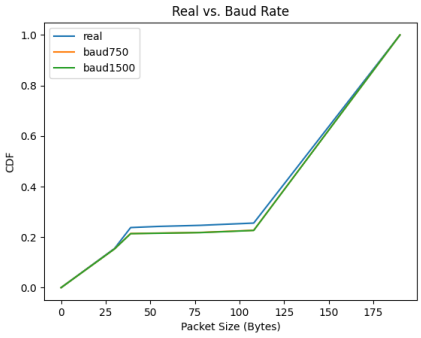

Figure 15: The packet size CDFs of Voiceover transmissions across different baud rates is shown. The CDF of real conversations is also shown as a comparison. We can see that the packet size CDFs for both baud rates closely matches that of the packet size CDF of a real conversation.

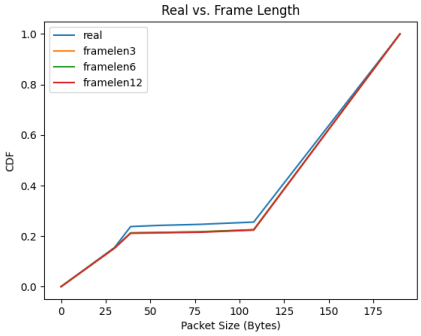

Figure 16: The packet size CDFs of Voiceover transmissions across different message frame lengths is shown. The CDF of real conversations is also shown as a comparison. We can see that the packet size CDFs for all three message frame length parameters closely matches that of the packet size CDF of a real conversation.

lengths easier in order to generate 30-second Voiceover transmissions. Regardless, we do not expect that a baud rate of 1599 symbols per second would generate different results. We collected roughly 1 hours worth of packet traces for each baud rate. The CDF of packet sizes for this experiment is shown in Figure 15.

C.4 Frame Length

In our message frame length comparison, we compared message frame lengths of 3, 6, and 12 characters. We fixed all other Voiceover modulation parameters to their default values and fixed the timing template used to shape the audio signal. The covert plaintext messages to be transmitted by Voiceover were randomly generated. Since the message frame length parameter affects the overall length of a Voiceover transmission due to per-frame overhead, the message lengths of the random plaintexts cannot be kept constant across the message frame length variables (each Voiceover transmission must be roughly 30 seconds long by our experimental design). However, we expect that differing message lengths will have no effect on the packet size distribution. We collected roughly 1 hours worth of packet traces for each message frame length. The CDF of packet sizes for this experiment is shown in Figure 16.

C.5 Bidirectional Communication

In this experiment, we aim to simulate real-world usage of Voiceover, in which two Voiceover endpoints are simultaneously communicating with each other using Voiceover-modulated audio transmissions. Audio modulation param-

eters were set to their defaults for both endpoints. The timing templates were randomly sampled from the GAN to simulate real usage of Voiceover. The covert plaintext messages to be transmitted by Voiceover were randomly generated, and we do not relax the duration of Voiceover transmissions - each Voiceover audio transmission is 5 minutes long. We collected roughly 4 hours worth of packet traces in this setting. The CDF of packet sizes for this experiment is shown in Figure 17.

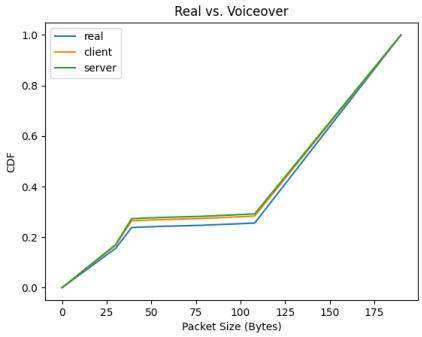

Figure 17: The packet size CDFs of real conversations and bidirectional Voiceover transmissions are shown. In bidirectional communication, one endpoint was arbitrarily set to be a client and the other was arbitrarily set to be the server. Each endpoint's corresponding packet size CDF was generated from the packet traces collected at that endpoint. We can see that the packet size CDFs of both directions of Voiceover transmission closely match that of the packet size CDF of a real conversation.