Facet: Streaming over Videoconferencing for Censorship Circumvention

Shuai Li
University of Minnesota
shuai@cs.umn.edu

Mike Schliep
University of Minnesota
schli116@umn.edu

Nick Hopper
University of Minnesota
hopper@cs.umn.edu

ABSTRACT

In this paper, we introduce *Facet*, an unobservable transport service for social video sites. Facet evades detection by Internet censors by streaming social videos over Skype calls, and applying a novel traffic-analysis countermeasure called video morphing. We report on the performance and security of a prototype implementation of Facet and find that a single Facet server can support roughly 20 simultaneous sessions, while providing strong unobservability: using the best known traffic analysis methods, a censor seeking to block 90% of Facet calls would need to block over 40% of all Skype calls. An additional benefit of our prototype implementation is that it avoids the distribution problem: clients can use Facet without installing any additional software.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

Keywords

censorship resistance, video conferencing, traffic analysis

1. INTRODUCTION

As the Internet has become a more useful tool for communicating information between individuals, censors working for Nation State Adversaries have responded by blocking access to the unfiltered versions of these sites. Additionally, these censors have also deployed increasingly sophisticated tools to identify and block access to the tools designed to circumvent this censorship, such as Tor [13] and other proxy services. These tools include sophisticated protocol fingerprinting via deep packet inspection (DPI), and even active probing attacks, in which suspected relays are contacted by the censors in order to confirm participation in the Tor protocol.

In response, researchers have developed systems that attempt to provide proxy steganography, which intend to make proxy connections resemble innocent "cover" protocols. For example, "decoy routing" [16, 20, 31] systems make connections to relays resemble TLS connections to random websites by hiding the relays in routers; SkypeMorph [22], CensorSpoofer [26], StegoTorus [27] and FreeWave [17] attempt to make proxy connections look like VoIP calls; and Collage [12] hides information in photos posted to content-sharing sites such as Flickr.

However, recent research [14, 15, 24] has revealed that these systems fail against more sophisticated censors due to several different inconsistencies between the proxy protocol and the cover protocol:

- *Emulation* inconsistencies can occur because either the client or proxy does not perfectly implement the cover protocol. StegoTorus' HTTP module does not respond to requests in the same way as any well-known HTTP server, and SkypeMorph does not simulate the TCP control connections [15].
- *Channel* inconsistencies can occur because the cover protocol responds differently to channel behavior than the proxy protocol, allowing an adversary to disrupt proxy connections while minimally impacting innocent connections. Decoy routing implementations can fail if the censor distributes packets across multiple AS paths to the "overt" destination, whereas TLS connections will not be affected [24]; VoIP and videoconferencing protocols are typically loss tolerant whereas proxies fetch general data and cannot tolerate packet loss [14].
- *Content* inconsistencies can arise when the behavior of a cover protocol depends on the characteristics of the traffic it carries and proxies do not match content to these characteristics; for example, since the FreeWave server tunnels modem traffic instead of voice signal over a VoIP channel, its communication session can be identified by traffic analysis [14, 28–30].

In light of these potential problems, finding a single cover protocol to carry arbitrary Internet content seems difficult. However, a recent survey of Chinese users of circumvention tools [1] found most users circumvent the Chinese "Great Firewall" to use three services: unfiltered search engines such as Google, uncensored social networks such as Facebook and Twitter, and video sharing sites like YouTube and Vine. This raises the possibility of serving most circumvention needs through a small set of unobservable transports.

In this paper, we present Facet, a system that enables the clients in a censored regime to watch YouTube, Vine and Vimeo videos in real-time. The basic idea of Facet is to send videos from these sites as the video content of a videoconferencing call – in the case of our prototype, a Skype call – between a Facet server and a client. Like all proxy steganography systems, it relies on the assumption that the censor is unwilling to indiscriminately block all or most sessions of the cover protocol (Skype) to avoid "collateral damage". Under this assumption, Facet provides the following features:

This work was supported by NSF under grant 1314637.

WPES'14, November 3, 2014, Scottsdale, Arizona, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2948-7/14/11 ...$15.00.
http://dx.doi.org/10.1145/2665943.2665944.

- Facet is **Emulation Consistent:** because the video is transmitted over an actual two-way Skype call, there is no difference between implementations to allow identification.
- Facet is **Channel Consistent:** we transmit videos over a channel intended for videos, so any disruption to a Facet session would cause the same disruption to a regular call.
- Facet is **Content Consistent:** Arbitrary videos may have different characteristics from videoconferencing calls, leading to detectable differences in packet sizes. We implement a binary classifier similar to the approach from Wright *et al.* [30] and show that unaltered YouTube videos sent over Skype are distinguishable from Skype calls. To defeat this, we introduce "video morphing," in which the Facet server frames the requested video within a randomly selected videoconference call. This increases the false positive rate of a classifier that can recognize 90% of Facet calls to nearly 40%.
- As a result, Facet provides **Unobservability**. Since videoconferencing streams are encrypted in transmission, it is difficult for censors to detect Facet sessions. Even if the videoconferencing software (or servers, in case calls are not routed directly between peers) is compromised, the use of randomized video morphing forces the censor to decode and analyze all video calls in real-time to detect Facet sessions.
- Facet's throughput is high enough to provide **real-time video delivery.** Most steganographic anti-censorship tools are designed for regular web browsing, and often have limited bandwidth for clients. In contrast, Facet is aimed at delivering real-time video service for clients, and achieves the same throughput as the videoconferencing service.
- Our approach is **provider independent.** Since the emulator devices in Facet are built independently from the videoconferencing systems, Facet can be adopted widely on any conferencing platform, such as Google Hangout, Skype, FaceTime or QQ. This feature not only provides accessibility to users who have access to different videoconferencing systems, it also provides the ability to evade blocking targeted at a single protocol or implementation.
- **No deployment at client side.** For Facet clients, there is no need to install any client software (which is often blocked), or to pre-share secrets with the server. This property makes Facet easier to use and maintain, since software updates only need to be applied by servers outside of the censored region.

We built a proof-of-concept implementation based on Skype videoconferencing service, and tested it in a real-world environment.

The remainder of the paper is organized as follows. Section 2 clarifies the threat model and design goals, and Section 3 gives related works. In Section 4, the high-level design of Facet is given, and Section 5 describes our prototype implementation. In Section 6, attacks by traffic analysis are introduced, and the countermeasures – video & audio morphing – are described in Section 7. We analyze the security of the Facet in Section 8, and give experimental results in Section 9. Section 10 concludes the paper.

2. THREAT MODEL AND DESIGN GOALS

2.1 Threat Model

We assume a state-level censor seeking to block unwanted Internet connections and the usage of circumvention tools. Specifically, the censor is considered to have the following capabilities:

In-depth Traffic Analysis. Censor can sniff the suspicious traffic and block it if signs of unwanted connections are found. The techniques can be keyword filtering, static IP address filtering, and protocol fingerprinting. Also, the censor is considered capable of analyzing covert traffic by statistical techniques. Recent studies [28–30] show the packet length in covert protocols leaks information about the transmitted traffic, and it can be used by the censor to infer the usage of circumvention tools.

Proactive Detection. The censor can be proactive. It can act as a user of circumvention tools for blockage. This attack can be proactive probing or enumeration attacks. For probing, the censor sends probes to potential circumvention proxies, which will be blocked if responding to provide the circumvention service. Enumeration attacks refer to the censor enumerating a proxy pool list, manually or automatically, for blocking these proxies. Both of these two attacks have been shown in real-world censorship.

Active Interference. The censor can interrupt circumvention tools by actively interfering. Particularly, the censor can delay, drop, or even inject packets into the session of potential circumvention systems. Such active interference, if crafted properly, does not necessarily increases the false alarms. For example, Geddes *et al.* [14] show the censor can drop Acks in Skype sessions to disrupt SkypeMorph with little interference to the genuine Skype sessions.

The censor is considered to permit widely-used encrypted protocols such as TLS and IPsec. In addition, the censor is assumed unwilling to block all videoconferencing systems, such as Skype, Google Hangout, and QQ. The reason could be business related or the political costs of doing so. This assumption is supported by the unblocking of GoAgent, one of the most popular circumvention systems in China. It is believed that the concern of the government about the side effects on its economy keeps GoAgent alive [1].

2.2 Design Goals

Consistency. The Facet design should satisfy the consistency principle. It should be consistent with the genuine system in terms of protocol, architecture, and content.
Unblockable. Consistency alone is not sufficient to make Facet unblockable. In addition, Facet should be designed to defend against traffic analysis, denial of service attacks, and active interference.
Scalability. The design of Facet should be independent from the specific videoconferencing systems. This scalability will make Facet applicable to a wide range of videoconferencing systems. This property not only makes Facet accessible for the client to use, but also makes it difficult for the censor to block.
Real-time Delivery. Facet should provide real-time delivery.

3. BACKGROUND AND RELATED WORKS

3.1 Videoconferencing Systems

To guarantee real-time delivery of loss-tolerant content, videoconferencing systems usually adopt UDP [2,6] to transmit encrypted video/audio packets. The architecture could be peer to peer, in which the traffic is transmitted between clients. Skype belongs to this category [10]. It could also be centralized, which means the traffic between clients is relayed by a central server. Google Hangout and FaceTime lie in this category [4]. Codecs are used to encode and decode digital data streams, with most systems adopting variable bit rate (VBR) encoding for coding efficiency. However, VBR has been proven to leak information about the transmitted content [28–30]. These studies show by traffic analysis, the attacker can determine which language is spoken in a Skype VoIP session with high accuracy.

3.2 Parrot Circumvention Systems

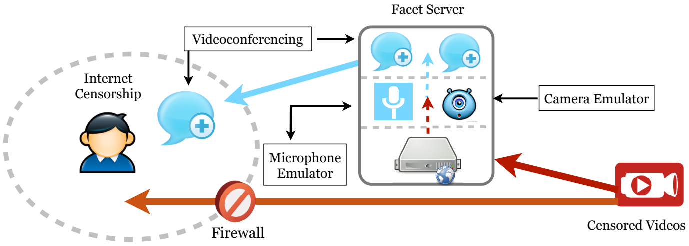


Figure 1: Facet System Architecture

SkypeMorph [22] disguises the connection between a Tor client and bridge as Skype call traffic. It initiates the Skype video call between the bridge and client as camouflage, then it will *drop* the genuine Skype connection and transmit Tor's TCP traffic over non-Skype UDP. A packetizer module is used to make the UDP traffic indistinguishable from Skype UDP traffic under traffic analysis.

StegoTorus [27] tries to obfuscate the Tor protocol. When choosing HTTP request as a cover protocol, it embeds hiddentexts in the URL and cookie fields. For HTTP responses, attached files such as PDF and Flash are used to carry the hiddentexts.

CensorSpoofer [26] provides an unblocked web browsing service by IP spoofing. Since upstream traffic (the requested URLs) is lightweight, it uses low capacity channels like emails for transmission. For downstream traffic, which is heavy HTTP responses, it directly sends the traffic to the user, but with the IP source address faked to fool the censor. Thus, the IP address of the proxy will not be revealed in both upstream and downstream traffic.

FreeWave [17] hijacks the Skype protocol for censorship circumvention. It uses Skype acoustic channels to transmit the web browsing traffic between a FreeWave client and server. The client modulates the web request into an acoustic signal, which will be carried by the VoIP service. At the FreeWave server side, it will extract the request by demodulating the acoustic signal, and proxy it to the blocked website. Since FreeWave directly uses the genuine Skype VoIP service, it is claimed to avoid flaws in camouflage and therefore be unobservable.

3.3 Attacks on Circumvention Systems

Parrot systems fail to achieve the claimed security goals due to the inevitable imitation flaws and the intrinsic inconsistency between the genuine and the circumvention protocol.

Imitation Flaws. Though these systems are aimed at providing unobservable proxies, a recent study [15] shows they fail to fulfill this goal. The unobservability of these systems requires the perfect imitation of the genuine systems. Unfortunately, the complexity and proprietary nature of the genuine system make this task quite challenging, and the imitation flaws can be exploited by the censor to detect camouflage systems. The discovered flaws include:

Fail to imitate side channels. The genuine systems usually have side channels for traffic control, user login, etc. However both SkypeMorph and StegoTorus fail to implement these channels.

Fail to imitate reactions. The camouflage systems behave differently than genuine systems in handling errors and network conditions. StegoTorus returns different error messages when the censor sends HTTP requests to it. CensorSpoofer fails to properly select the spoofed IP address/port, making the address/port behave differently from a genuine SIP client in responding to a censor's probe.

Incorrect in imitation. Both SkypeMorph and StegoTorus wrongly imitate Skype UDP packets for lack of SoM field. Also StegoTorus generates incorrect PDF lacking the *xref table*. In addition, both systems reuse generated trace, failing to represent Skype traffic.

Inevitable Inconsistency. Recent work [14] reveals even perfect emulation can not guarantee the proxy unobservability and unblockability. The failure roots in the inevitable inconsistencies between genuine and proxy protocols in terms of content and channel.

Content Inconsistency. In FreeWave, a modulated acoustic signal rather than human speech is transmitted over VoIP. This content inconsistency is proved sufficient for a censor to identify FreeWave traffic. Since VoIP protocols usually adopt VBR encoding, the packet length can reveal the transmitted content. Thus a censor can distinguish and block FreeWave traffic by traffic analysis.

Channel Inconsistency. Both SkypeMorph and FreeWave require reliable channels to transmit Tor TCP packets or synchronization frames. But VoIP usually adopt unreliable UDP to transmit audio/video which can tolerate packet losses. This inconsistency enables a censor to interrupt SkypeMorph or FreeWave by packet dropping, while having negligible impacts on the genuine VoIP service. It is shown that dropping 90% of the packets for less than a second can desynchronize the FreeWave Modem, and by dropping only 5% of packets SkypeMorph can be stalled indefinitely.

4. THE FACET DESIGN

Facet delivers censored videos in real-time. As is shown in Figure 1, the procedure of a Facet connection is:

1. The Facet server distributes its conferencing ID for service discovery. The distribution can be public or private, depending on the architecture of the videoconferencing system.
2. A Facet client sends a contact request to the server. After the server accepts the request, they establish initial connections.
3. The client sends the Uniform Resource Locator of the censored video to the server by an instant message or an email.
4. The server extracts the client's request, initiates audio & video emulators, and places a conferencing call to the client.
5. Simultaneously, the server forwards the URL to the Facet pipeline, which will download, decode, and resize the requested video, finally streaming it into the emulator devices.
6. After accepting the videoconferencing request, the client can watch the video in the videoconferencing session.
7. The client can also send control commands to the server for video playback or adjusting video speed.
8. After the video is over, or the client ends the conferencing, the Facet server destructs the emulators, and ends the session.

Navigation. For the Facet clients, an important question is how to navigate and discover the URL of the censored video. There are three methods for discovery.

Encrypted Video Search. For regimes where encrypted web search services (provided by Google, etc.) are not blocked, the client can use such a service for navigation. The client can specify the keywords and websites for video searching, and the search engine will return the results with the URLs of the videos.

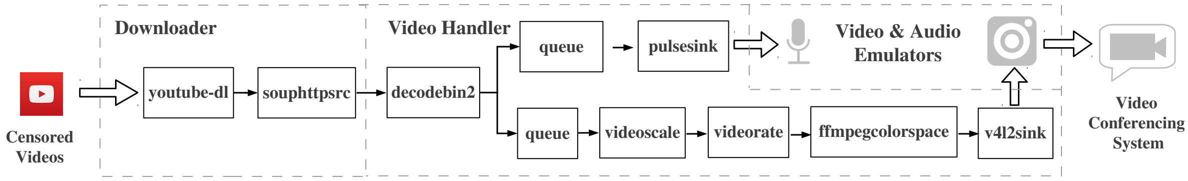


Figure 2: Facet Pipeline: to deliver censored videos in real-time

Video Subscription. The client can also use a subscription service. For websites such as YouTube, the client can make a subscription to the videos, and periodically it will receive emails including subscribed video information, such as URLs and titles.

Search Engine Proxy. Facet implementation also includes a search engine proxy using email tunnels. The client can email search keywords to the Facet server, which will fetch the search results, and email a screenshot of the page back to the client. Then the client can find the videos to watch and their URLs. The email address of the Facet server can be publicly distributed, and the unobservability of the email tunnels guarantees its security [32].

Service Discovery. The Facet server has two strategies to distribute its conferencing ID.

Public Distribution. For centralized videoconferencing systems, such as Google Hangout and FaceTime, the Facet conferencing ID can be publicly distributed. Though this strategy also discloses the ID to the censor, it does not increase the censor's ability to block the service. Since the videoconferencing traffic is encrypted the censor can not link the conferencing ID to a specific session to block. Even though the censor may proactively probe the service, it can not pinpoint the Facet server IP address, because this address is hidden behind the videoconferencing server. Thus, the censor's ability to distinguish and block the Facet session is not improved, even when it knows the server's conferencing ID.

Private Distribution. For decentralized videoconferencing systems such as Skype, the two entities send traffic to each other directly. Consequently, the Facet server ID should only be distributed privately. Otherwise, the censor can pinpoint and block the Facet server IP address by proactively probing the service.

Security. The censor may block the potential Facet session in which the video stream is only unidirectional. In this situation, the Facet client should enable the camera in the conferencing. To prevent denial-of-service (DoS) attacks, the Facet server is configured to not accept strangers' requests. Thus, a potential Facet client is required to register with the server, by sending an "add contact" request to the server's conferencing ID. Only after this request is proved by the Facet server, can the client access to the service. Also, the Facet server enforces usage limits on each registered client ID to further defend against DoS attacks.

5. IMPLEMENTATION

The Facet server is implemented by selecting Skype as the videoconferencing system. The server is built on Ubuntu 12.04 Precise Pangolin, and can support most popular video sites, such as YouTube, Vine, and Vimeo.

5.1 Facet Pipeline

The real-time delivery of the censored videos requires Facet to construct a pipeline to handle video downloading, decoding, and playing in parallel. The pipeline implementation mainly relies on Gstreamer [3], an open source multimedia framework. A typical Facet pipeline is shown in Figure 2, which consists of the Downloader, Video Handler, and Camera & Microphone Emulators.

Downloader. Popular video websites usually utilize HTTP based dynamic video streaming [18], so the downloader needs to decode the video URL to obtain the actual data streaming address. This address is obtained by utilizing `youtube-dl`, which is an open source video downloading toolkit, and can support websites such as YouTube, Vimeo, Vine, and MetaCafe. Then, this obtained streaming address is forwarded to a Gstreamer element `souphttpsrc` to download the video. It is worth noting that `souphttpsrc` can directly forward the received stream to the video handler, without having to wait until downloading the entire video.

Video Handler. The video handler is implemented with the Gstreamer framework to convert the downloaded stream live. The functionality of Gstreamer elements is listed in Table 1. The stream is split by `decodebin2` to obtain the video & audio stream. Each is placed into its respective emulator for playing. Since the downloaded video stream may fail to satisfy the emulator's requirement in colorspace, resolution, and frame rate, additional elements such as `videoscale`, `videorate`, and `ffmpegcolorspace` are utilized to make the conversion.

Emulator. Facet initiates two emulators to deliver the video & audio stream. For the camera emulator, our Facet implementation utilizes `v4l2loopback`, a kernel module to create a v4l2 device emulator. For the microphone emulator, Facet utilizes `pactl`, a program controlling PulseAudio sound server, to initiate a microphone device instance. Both of these two emulators can be recognized by the conferencing systems.

5.2 Other Implementation Details

The Facet server needs to initiate or end videoconferencing request automatically. For our implementation, we use `skype4py` [8] for automation, which is a python wrapper for the Skype API. Also, the server can run multiple videoconferencing sessions to serve more clients. Our implementation shows for a Facet server which has 15 Mbit/s bandwidth and 4 virtual cores, it can support up to 20 simultaneous sessions.

Another implementation detail is URL submission. Facet supports instant message submission: the user can give the video URL to server by using the videoconferencing's instant message service. Facet also supports email submission, which is more secure when the videoconferencing provider is considered to collude with the censor. A detailed analysis is given in Section 8.

6. TRAFFIC ANALYSIS

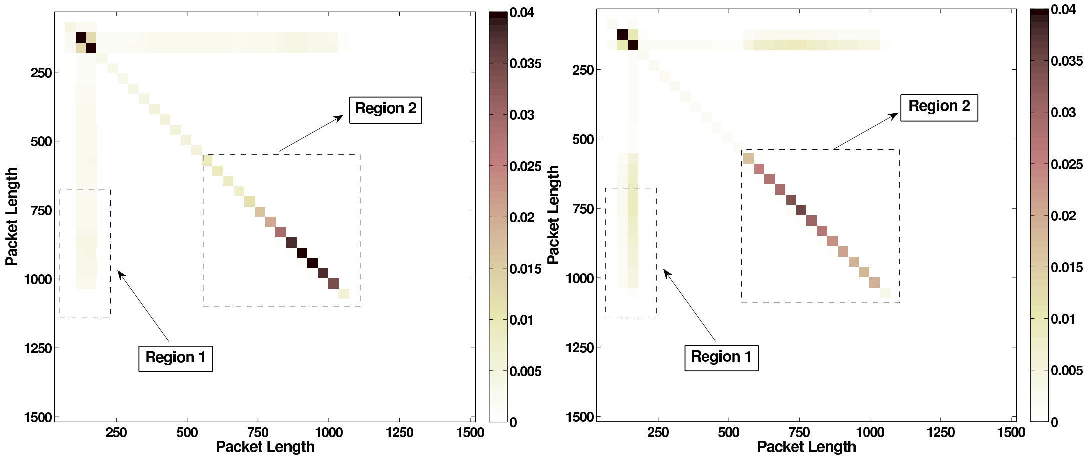
Since videoconferencing adopts VBR codecs and length preserving encryption, the packet length can leak information about the content being transmitted. Previous research shows the rate of distinguishing phrases, languages, and even speaker identity in a VoIP conversation. Thus, it is necessary to study whether the censor can distinguish the Facet connection by traffic analysis.

Classifier. [30] reveals a χ^2 classifier to distinguish the language in a VoIP call. With about 90% accuracy, the classifier can

Facet adopts Google video search as the search engine, and the video URLs are literally shown in the screenshots.

Element	Fuctionality
souphttpsrc	receive HTTP network data as a libsoup client
decodebin2	decode and demultiplex the data stream from souphttpsrc
queue	audio/video stream queue
videoscale	resize the received video frame to match the camera emulator
videorate	manipulate the timestamps on video frames to adjust frame rate
pulsesink	direct audio to the PulseAudio server
v4l2sink	play the video in v4l2 device
ffmpegcolorspace	convert the video from one colorspace to another

Table 1: Gstreamer Elements



(a) 2-gram Distribution for Chat Videos

(b) 2-gram Distribution for YouTube Videos

Figure 3: Traffic Analysis: Chat and YouTube videos have different traffic patterns.

narrow down from the two possible languages to one. We adopt this best known binary classifier, and investigate whether a censor can identify the Facet session, or in other words, how accurate it can determine whether the videoconferencing is genuine or not.

The χ^2 classifier takes the packet length as input, and adopts n-gram as feature extraction, which is a contiguous sequence of n packet lengths from the time series traffic. Suppose the traffic is (a, b, c, d), where a, b, c, and d are the packet lengths, then the 2-grams are (a, b), (b, c), and (c, d). The reason for not including traffic delay in the classifier is that this feature is not stable. It can be easily affected by network conditions. Besides, the delay in VoIP traffic is usually fixed [29]. The packet length is discretized into equal partitions of size K before calculating the n-gram, to avoid the curse of dimensionality and improve the classification accuracy. $G_K(n)$ is used to denote the set of all the possible discretized n-grams. Then, for a given traffic, the probability over each element of $G_K(n)$ can be used as its fingerprint for classification.

Training. Let T_0 denote the set of genuine chat videos, and T_1 denotes the set of the censored videos in the training process. The models for the genuine chat and censored videos are built as follows:

$$\bar{Pr}(i,g) = \frac{1}{\sum_{v\in T_i} N_v} * \sum_{v\in T_i} N_v * Pr(i,v,g),\ g \in G_K(n) \quad (1)$$

where N_v denotes the number of grams for video v.

Classifying. The original classifier assigns a test case to the category whose model is closer to that of the test case in terms of χ^2 distance. This decision rule only allows fixed false positive/negative rates. Considering a censor should be able to adjust its aggressiveness in blocking unwanted connections, we use the following rule:

$$\Theta = \frac{\Delta(v, T_0, G_K(n))}{\Delta(v, T_1, G_K(n))} \quad {}^{\mathcal{H}_0}{<}_{\geq \mathcal{H}_1} \quad \delta \qquad (2)$$

where $\mathcal{H}_0$ represents the video is classified as chat video, and $\mathcal{H}_1$ denotes it is determined to be a censored video. Θ is the χ^2 distance ratio, and $\Delta(v, T_i, G_K(n))$ is the χ^2 distance between video v and training set T_i:

$$\Delta(v, T_i, G_K(n)) = \sum_{g\in G_K(n)} \frac{[\bar{Pr}(i,g) - Pr(i,v,g)]^2}{Pr(i,v,g)} \qquad (3)$$

This allows the censor to adjust the value of δ to change its blocking strategy. In the experiment part we will discuss how the censor chooses a proper δ value in order to block a given percentage of unwanted traffic. In addition, the classifier includes the gram selection algorithm introduced in [30]. The general idea is to exclude the grams which have negative influence on classification results.

For our dataset, we use the 2 fold cross-validation in evaluation [21]. The dataset is randomly separated into two groups d_0 and d_1 with equal size. Each group will alternatively be the training set in two rounds.

Accuracy. The censored video dataset consists of 1013 popular YouTube videos, and 1045 YouNow videos are used as the chat video set (details are given in section 9). With $n = 2$ and $K = 50$ the experimental results are shown in Figure 3 and Figure 4. The classification is measured by false positive rate, the probability of wrongly determining the traffic of chat videos as that of YouTube videos, and the false negative rate, the probability of determining the traffic of YouTube videos as that of chat videos [25]. It shows with a rate of 90%, the censor can correctly distinguish the traffic pattern of the streamed YouTube video, with only the cost of 10% false positive rate. Another observation is that the censor can even adjust needs to disrupt only 2% of genuine videoconferencing connections to block 80% of Facet connections. This result is further demonstrated by the difference on averaged traffic patterns between chat videos and YouTube videos in Figure 3(a) and 3(b). Thus, traffic morphing is necessary to protect Facet from blockage.

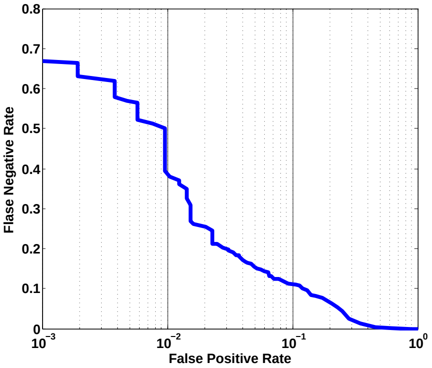


Figure 4: Traffic Analysis: the receiver operating characteristic (ROC) curve of the classifier

7. MORPHING

This section introduces video & audio morphing to defend against the censor's traffic analysis. An intuitive method for traffic shaping is to manipulate packets, but [15] demonstrated the flaws in such morphing. Facet proposes video & audio morphing to shape the traffic. The property of video & audio is transformed to simulate that of chat video & audio, modifying the traffic pattern without packet dropping.

7.1 Audio Morphing

Observation. Region 1 in Figure 3 reveals for genuine conferencing, there are more packets with a length in the range of 100 to 150 bytes and less packets with a length in the range of 150 to 200 bytes as compared to Facet streaming the YouTube videos. The primary reason for this difference is that the conferencing audio has lower quality, and shorter packets can be used to transmit the audio. This difference is further demonstrated by the discrepancy in sampling rate. The relation between audio quality and sampling rate is that the higher the sampling rate, the better the quality [5]. For our chat video dataset, the sampling rate is 11,500hz, but a typical YouTube video has a sampling rate of 44,100hz. In addition, the chat videos have only one channel with 16 bit width, while YouTube videos have two channels with 32 bit width.

Audio Morphing. The Facet audio is resampled live to simulate the quality of the chat audio, and this requires Facet pipeline to include a `audioresample` element. In the following, we take the YouTube audio as an example, and show how to determine the resampling rate empirically. For the chat video dataset, the video & audio is streamed simultaneously, while for the YouTube dataset, YouTube audio and chat video are streamed. Thus, the difference in traffic pattern is only related with the audio. Figure 5 (a) shows when the sampling rate is 3,000, the morphing is at its best. Also, the YouTube audio which usually has two channels is converted into mono-channel, and the bit width is set to 16.

It is worth noting that though the classifier can still do better than random guessing, the audio morphing can *practically* disable the detection as is analyzed in Section 8.

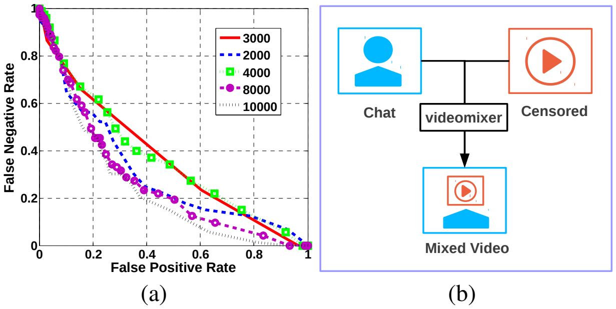


Figure 5: (a) Audio Morphing: choose resampling rate, and (b) Video Morphing: embed the censored video in a chat video

7.2 Video Morphing

Observation. Region 2 of Figure 3 shows chat video traffic has more large packets than that of YouTube videos. Our results show chat videos have more packets in the range of 400 to 600 bytes, and 800 to 1000 bytes than YouTube videos. The possible explanation is that the chat videos are usually slower motion which causes the video encoder to handle them differently.

The encoder can use the temporal redundancy of the slow motion video to optimize its coding efficiency. Take the H.264 codec as an example. There are mainly three types of frames in H.264: the I frame, B frame, and P frame [7, 19]. The I frame is encoded independently from other frames, only exploiting the spatial redundancy to compress the video stream. Differently, the P frame and B frame depend on the previous frames or even future frames, taking advantage of temporal redundancy. Typically, the I frame is significantly larger than the P and B frames. Thus, for videos with different temporal redundancy, the encoding results are different.

Video Morphing. The video morphing in Facet is based on the block-oriented feature of H.264 codecs. In H.264, each frame is divided into multiple small square blocks called macroblocks. The coding tools or kernels are applied to these macroblocks rather than the whole frame. Facet performs its traffic shaping by manipulating these macroblocks. Specifically, it places the censored video on several adjacent blocks, with a randomly selected chat video being played on the remainder of the blocks. By adjusting the width and height of the censored video, the traffic pattern is expected to shift between that of the censored video and the chat video. This mechanism is illustrated in Figure 5 (b).

The Facet server can make the trade off between steganography and the video quality. We define the steganography level s to be the scale of the censored video. Suppose the original width and height for the video is w and h, then the embedded video has width $w \cdot s$ and height $h \cdot s$. The experimental results in Section 9 show the video morphing can effectively defeat the censor's traffic analysis.

8. SECURITY ANALYSIS

Attacks Exploiting Emulation Flaws. Facet uses the videoconferencing systems directly without trying to emulate the system, the emulation flaw problem does not exist. So, such attacks fail to block Facet connections.

Attacks Exploiting Content Mismatch. Facet is content consistent. It utilizes the video (or audio) channel to transmit the video (or audio). In addition, the morphing techniques can further secure Facet against the traffic analysis. Our experimental results show the censor is incapable of blocking the Facet without *apparently* disrupting genuine connections. Considering the assumption that the censor is not willing to largely disrupt the videoconferencing systems, our morphing techniques can *practically* disable traffic analysis. This is further supported by the observation that genuine videoconferencing connections are enormously more frequent than Facet connections, which means if the censor tries to block Facet, the number of blocked genuine videoconferencing connections would be much more than the blocked Facet connections.

Attacks Exploiting Architecture Mismatch. When a centralized conferencing system is adopted, its server relays the conferencing traffic between two clients. This is consistent with the proxy-client architecture, and the censor can not detect Facet connections by architecture analysis. If Facet utilizes a decentralized conferencing system, The Facet server can tunnel through multiple proxies out of a censored region when connecting to clients via videoconferencing. This strategy can prevent Facet server from being the hot spot, and secure it against the architecture analysis.

Attacks Exploiting Channel Mismatch. Facet is channel consistent. Since the nature of its channel is exactly that of the genuine channel, the packet dropping, duplicating, and delaying will not have any more impact on Facet than the videoconferencing system. Thus, Facet is secure against attacks exploiting channel mismatch.

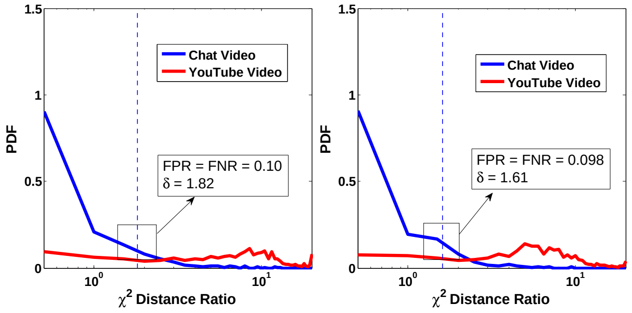
Private Watching. A corrupted Facet server may pose as a potential privacy violation to the client. The strategy for the client to protect its privacy is to use a pseudonym ID. Then, the Facet server can only link the watching history to a pseudonym rather than the client. The server may try to break such pseudonym by IP geolocation. We argue that if the videoconferencing system is centralized, the client's actual IP address is hidden behind the central server, thus the Facet server can not know the client's IP address. For decentralized videoconferencing systems, the client should configure a proxy before initiating the videoconferencing to defeat this attack. This proxy can be located in or out of the censored region.

Denial of Service Attack. When Facet server ID is distributed publicly, the censor may launch a DoS attack. CAPTCHA [11] can be used to mitigate the censor's enumeration. In addition, the server can enforce usage limitations on each client ID. Besides, puzzles can be used to defeat the Sibil identify attack.

Colluding Videoconferencing Provider. A provider may collude with the censor. In such situation, the client should submit its URL by emails, not instant messages, or the provider can detect the Facet connection by text surveillance. We argue even if a provider is colluding with the censor, it is much more expensive and difficult to filter audio/video than text messages. In addition, the provider has to block the connection quickly, or the blockage is too late to have a significant effect. Furthermore, the number of videoconferencing connections could be too large for the provider to check in time. Thus, the provider is believed unlikely to block Facet connections. Also, since Facet is independent from any particular provider, we can simply turn to the other providers which do not collude with the censor to avoid such attacks.

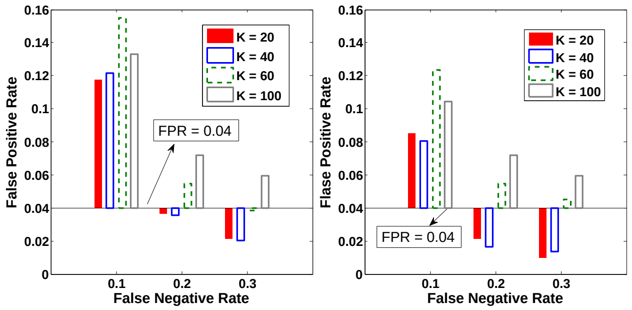
Fingerprints of Frequently Viewed Videos. Popular videos may be viewed frequently by Facet users, resulting in the unchanged replay of the traffic pattern. The censor may identify this repeated pattern and block the session. In order to combat this attack, the Facet server can switch the background chat video or modify the audio rate in each session to obfuscate the traffic fingerprint. Considering the Facet server has plenty of choices in selecting the background chat video, this scheme can provide sufficient obfuscations. Another possible fingerprint of the frequently viewed videos comes to the video length. To obfuscate it, a random video can be appended right after the requested video. Then for the censor, it can not identify the repeated sessions by video length.

9. EXPERIMENT



(e) $K = 20, n = 2$ (f) $K = 20, n = 3$

Figure 6: YouTube Video without Morphing: probability density function of χ^2 distance ratio Θ



(a) $n = 2$ (b) $n = 3$

Figure 7: False Positive Rate (no Morphing): if the censor blocks 70%, 80%, or 90% Facet connections.

9.1 Dataset

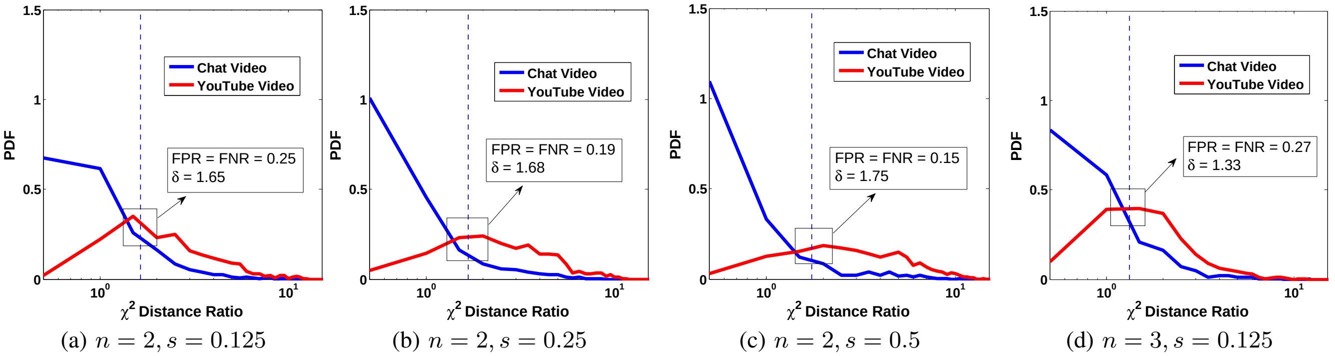
Chat Video Set. The chat videos are from YouNow.com, a popular live video blogging website. YouNow videos can simulate the chat video well for the following reasons. First, it does not lack interactions. A special feature of YouNow is that it allows the audiences to interact with the blogger by instant messages. Consequently, the blogger often pauses and answers questions, simulating the interactions in genuine video chat. Second, the users seldom use any video editing in the live video blogging (the reason could be the inconvenience of live editing), the videos hold the nature of webcam video chats. Third, the YouNow has a large amount of diversified webcam videos representing different situations in video chats. All these features make YouNow videos suitable.

The videos were collected on Sep 17, 24, and Oct 5. The users were picked out by using YouNow's "recently Broadcasted" feature, and for each user only their latest video is included. Finally, 1045 YouNow videos are included in the chat video set.

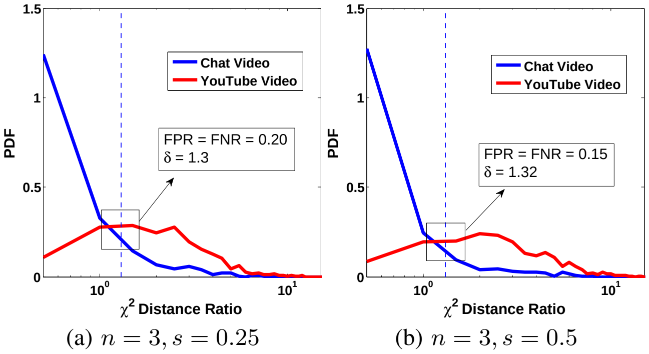
Censored Video Set. Highly viewed YouTube videos are chosen to construct the censored video set. YouTube charts list the most popular videos by category for a time period of one week, one month, or all time. For each category, we harvested all the videos with length more than 1.5 minutes. 1013 YouTube videos under 15 categories are included in the dataset, as is shown in Table 2. The video format is selected to be FLV, and the resolution is 360p.

9.2 Experimental Setup

The experiment is run on a MacBook Pro with a 2.3Ghz Intel Core i7 processor, and OS X version 10.9. A guest operating system, Ubuntu Precise Pangolin (12.04), is installed on a VMware virtual machine, in which the Facet server is implemented. The



(a) $n = 2, s = 0.125$ (b) $n = 2, s = 0.25$ (c) $n = 2, s = 0.5$ (d) $n = 3, s = 0.125$



(a) $n = 3, s = 0.25$ (b) $n = 3, s = 0.5$

Figure 8: YouTube Video with Morphing: probability density function of χ^2 distance ratio Θ

host machine acts as the Facet client. The connection between host and guest machine is through a private virtual network [9].

Skype is selected as the videoconferencing system. The traffic between the two videoconferencing clients is exchanged through the VMware private virtual network. For the host, the Skype version number is 6.9 (701), and for the guest machine, the version is 4.2.0.11. Skype uses H.264 as the video codec, and SILK_V3 as the audio codec. The Frames Per Second (FPS) is 30.

Both of chat videos and YouTube videos are streamed into camera & microphone emulators. For each video, the packet capture is started after the video has been playing for 30 seconds, and the capture lasts for one minute. Skype's technical call information shows the packet loss rate is 0% in the experiment. For the camera emulator, we set the timeout to 1000 seconds, with YUY2 format. The emulator resolution is selected to be 320 × 240.

Category	No.	Category	No.	Category	No.
Animation	80	Autos	39	Travel	38
Comedy	85	Edu	84	Sci	51
Entertain	74	Gaming	98	Sport	59
Howto	86	Music	94	People	64
News	57	Nonprofit	58	Pet	46

Table 2: YouTube Video Set

9.3 Morphing Effectiveness

This part evaluates the effectiveness of Facet morphing. The experiment captures the traffic of the chat video set, morphed YouTube videos with $s = 0.125, 0.25$ and 0.5, and non-morphed YouTube videos. Then, the chat video set is grouped with each of these four YouTube video sets individually for binary classification.

We set $n = 2$ and $n = 3$ respectively. We do not include a larger n case, because the classifier with $n > 3$ performs poorly (and therefore not representative) due to the curse of dimensionality [23]. Suppose $n = 4$. If for each packet, there are 20 discretized lengths, then $20^4 = 160000$ grams are possible for videoconferencing traffic. When dimensionality is high, the volume of the space becomes so large that the the training samples are too sparse to represent the model, therefore, the classification in high dimensional space performs poorly. Also, we tune the discretization parameter K to be 20.

Figure 6 and Figure 8 give the probability density function (PDF) of the test statistic (χ^2 distance ratio Θ) for different cases, which indicates how the classifier performs for binary classification. Also, the false positive rate (or false negative rate) is given, when it equals to false negative rate (or false positive rate). This value can be used to compare the classifier performance for different cases.

Figure 6 shows the PDF of Θ when the classifier is used to distinguish genuine YouTube and chat video traffic. (a) uses 2-gram as the feature extraction, and (b) 3-gram. Both of them have $K = 20$. These figures show the traffic of chat and YouTube videos have distinct Θ distributions, thus the censor can specify a proper threshold to block a majority of unwanted traffic while keeping most of the genuine conferencing connections alive.

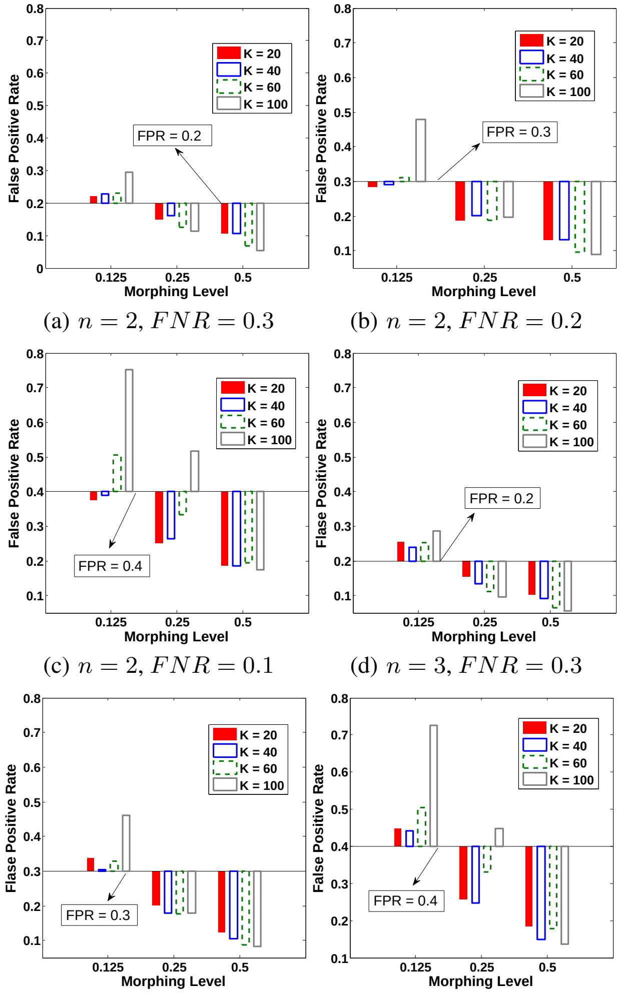
Figure 8 is the PDF of Θ when the classifier is used to determine morphed YouTube video traffic from chat video traffic with $K = 20$. For (a) and (d) $s = 0.125$, and the figures show Θ of the morphed YouTube video has more distribution in the 0 to 1 range, which means by morphing, the YouTube video is more likely to be regarded as the chat video. From the perspective of false positive/negative rate, the classifier only has $FPR = FNR = 0.25$ for 2-gram and $FPR = FNR = 0.27$ for 3-gram. The results demonstrate the morphing effectiveness.

Also, the distribution of Θ for different morphing levels is given in (b) (c) (e) (f). We show with a smaller morphing level s, the distribution of Θ for morphed YouTube videos resides more on the range of 0 to 1, and the false positive/negative rate is higher. This demonstrates when morphing level s is smaller, the morphed YouTube video session is more secure against traffic analysis.

9.4 Security Against Blockage

In our attack model, the censor is assumed to be unwilling to block or disrupt the genuine conferencing. Here, we investigate in order to block a given fraction of Facet connections, how likely the censor is to affect the genuine conferencing.

The censor is set to block $0.9, 0.8$ and 0.7 Facet connections. The corresponding false positive rate is investigated for morphed YouTube videos as well as the genuine YouTube videos. The experiment results are given in Figure 7 and Figure 9.



(a) $n = 2, FNR = 0.3$

(b) $n = 2, FNR = 0.2$

(c) $n = 2, FNR = 0.1$

(d) $n = 3, FNR = 0.3$

(e) $n = 3, FNR = 0.2$

(f) $n = 3, FNR = 0.1$

Figure 9: False Positive Rate (with Morphing): if the censor blocks 70%, 80%, or 90% Facet connections.

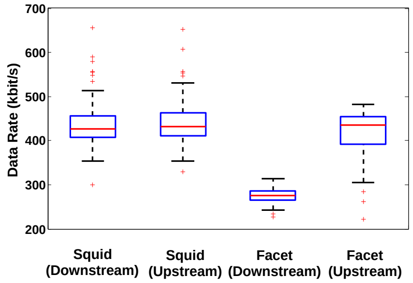


Figure 10: Bandwidth Consumption: Facet vs. Squid

	Downstream		Upstream	
	Skype	Gstreamer	Skype	Gstreamer
Mean	42.49	232.62	409.11	8.71
Max	51.83	265.32	471.91	10.81
Min	30.85	189.63	215.90	6.48
Med.	45.36	236.58	424.94	8.45
Std.	7.11	12.88	49.96	1.39

Table 3: Facet Traffic Break Down (kbit/s)

In Figure 7(a) and (b), the false positive rate is given when the censor tries to block 70%, 80%, or 90% Facet sessions playing genuine YouTube videos. We can see without morphing, the censor only has to disrupt 4% genuine videoconferencing to block 80% Facet connections. If it is aimed at blocking 70%, the false positive rate is even lower, only 2%. The figures show the necessity of adopting morphing mechanisms in the Facet design.

Figure 9 (a) to (f) show the false positive rate when the censor attempts to block 70%, 80%, or 90% Facet sessions playing morphed YouTube videos. For (a) and (d), the false negative rate is set to 0.3, and we can see even if the censor chooses the optimal parameters, such as $K = 20$ and $n = 2$, it has to disrupt more than 20% genuine videoconferencing. For (c) and (f), when the censor attempts to block 90% of the Facet connections, it has to block 40% of the genuine videoconferencing connections. To conclude, if the censor wants to massively block the Facet connections, it has to disrupt at least 20% genuine videoconferencing connections. This cost, under our assumption, can make Facet less vulnerable to blockage, especially considering genuine videoconferencing connections are enormously more frequent than those of Facet.

Also, the figures show the smaller s is, the more secure the Facet server is against blockage. But considering the aggressiveness and capability of the censor may vary, the value s can be adjusted by the Facet server to provide a higher quality service. In addition, it shows the classifier with $n = 2$ in general outperforms the classifier with $n = 3$, though 3-grams can have more complete feature extraction. The reason for this is the curse of dimensionality.

9.5 Performance Analysis

Setup. 106 YouTube videos (with length below 180s) are selected from our censored video dataset. When the Facet server plays these videos, the upstream & downstream bandwidth, together with CPU & Memory usage are recorded. Also the client is set to disable the camera. For the purpose of comparison, a traditional web proxy `Squid` is adopted. In the experiment, the client uses the Squid proxy and plays the YouTube videos by using the Firefox. Also, the video resolution is set to 240p in both cases.

Experimental Results. The bandwidth costs are given in Figure 10 and Table 3. It is shown that the downstream bandwidth of the Facet server is lower than that of the Squid server. If the client uses Facet, about 150 kbit/s downstream bandwidth is saved. A possible reason for this difference is that for web browsing, extra information (such as advertisements and pictures, etc.) are fetched, and for a Facet connection, the client only downloads the required video. When it comes to upstream, the bandwidth costs of these two systems are close. It is worth noting that although the downstream bandwidth of the Facet server will be increased if the client enables its camera, Table 3 shows the downstream bandwidth is still less than 700 kbit/s if the client's camera has the same resolution with that of the server. These experimental results show Facet server has high efficiency in bandwidth usage. For a Facet server with 15 Mbit/s bandwidth, it can support up to 20 simultaneous sessions.

The computational costs for these two systems are shown in Table 4. For Facet server, the costs comes from making a Skype video call (Skype), downloading and redirecting video & audio stream (Gstreamer), and executing a Skype wrapper (Python). Though the table shows the Facet server consumes more CPU cycles and memory than Squid (most of which comes from the Skype video call), this cost is acceptable. Still, a computer with one virtual core can support up to 5 Facet sessions, and for a computer with 4 virtual cores, it can support 20 sessions.

Category	Program	CPU	Memory (4GB)
Facet (s=1)	Skype	14.4%	2.4%
	Gstreamer	3.7%	0.5%
	Python	0.6%	0.1%
Web Proxy	Squid	0.4%	0.3%

Table 4: Facet CPU & Memory Usage

10. CONCLUSION

This paper presents Facet, a censorship circumvention system to deliver censored videos in real-time. Facet emulates the input devices for a conferencing system, and streams the censored video over them. Compared with other censorship circumvention systems, Facet is consistent with genuine conferencing in terms of content, channel, and architecture. To further defend against the censor's traffic analysis, audio and video morphing are proposed. Our experimental results show such morphing can effectively defend against traffic analysis.

11. REFERENCES

[1] Collateral freedom: A snapshot of chinese internet users circumventing censorship, https://openitp.org/news-events/collateral-freedom-a-snapshot-of-chinese-users-circumventing-censorship.html.

[2] A deeper look at skype's protocol, http://scenic.princeton.edu/network20q/wiki/index.php.

[3] gstreamer: open source multimedia framework, http://gstreamer.freedesktop.org.

[4] Hangouts faq for administrators https://support.google.com/a/answer/1261833?hl=en.

[5] http://music.columbia.edu/cmc/musicandcomputers /chapter2/02_05.php.

[6] Optimize your network for hangouts, https://support.google.com/a/answer/1279090?hl=en.

[7] Skype encoding camera specification, http://developer.skype.c om/resources/skype_encoding_camera_specification.pdf.

[8] skype4py, http://sourceforge.net/projects/skype4py/.

[9] Vmware virtual networking concepts, http://www.vmware.com /files/pdf/virtual_networking_concepts.pdf.

[10] What are p2p communications? https://support.skype.com/en /faq/fa10983/what-are-p2p-communications.

[11] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, 2003.

[12] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *Proceedings of USENIX Security'10*, 2010.

[13] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of USENIX Security'04*, 2004.

[14] J. Geddes, M. Schuchard, and N. Hopper. Cover your acks: Pitfalls of covert channel censorship circumvention. In *Proceedings of CCS'13*, 2013.

[15] A. Houmansadr, C. Brubaker, and V. Shmatikov. The parrot is dead: Observing unobservable network communications. In *Proceedings of IEEE S&P'13*, 2013.

[16] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of CCS'11*, 2011.

[17] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer. I Want my Voice to be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention. In *Proceedings of NDSS'13*, 2013.

[18] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of IMC'12*, 2012.

[19] B. Juurlink, M. Alvarez-Mesa, C. Chi, A. Azevedo, C. Meenderinck, and A. Ramirez. Understanding the application: An overview of the h.264 standard. In *Scalable Parallel Programming Applied to H.264/AVC Decoding*, SpringerBriefs in Computer Science, pages 5–15. Springer New York, 2012.

[20] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable internet communication. In *Proceedings of FOCI'11*, 2011.

[21] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1995.

[22] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. Skypemorph: Protocol obfuscation for Tor bridges. In *Proceedings of CCS'12*, 2012.

[23] J. Rust. Using randomization to break the curse of dimensionality. *Econometrica*, pages 487–516, 1997.

[24] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *Proceedings of CCS'12*, 2012.

[25] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[26] Q. Wang, X. Gong, G. T. K. Nguyen, A. Houmansadr, and N. Borisov. Censorspoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In *Proceedings of CCS'12*, 2012.

[27] 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 CCS'12*, 2012.

[28] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In *Proceedings of IEEE S&P'11*, 2011.

[29] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *Proceedings of IEEE S&P'08*, 2008.

[30] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob. In *Proceedings of USENIX Security'07*, 2007.

[31] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of USENIX Security'11*, 2011.

[32] W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov. Sweet: Serving the web by exploiting email tunnels. In *Proceedings of HotPETs'13*, 2013.