I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention

Amir Houmansadr† Thomas Riedl‡ Nikita Borisov‡ Andrew Singer‡
†The University of Texas at Austin
‡University of Illinois at Urbana-Champaign
amir@cs.utexas.edu {triedl2,nikita,acsinger}@illinois.edu

***Abstract*—Open communication over the Internet poses a serious threat to countries with repressive regimes, leading them to develop and deploy censorship mechanisms within their networks. Unfortunately, existing censorship circumvention systems face difficulties in providing *unobservable* communication with their clients; this highly limits their *availability* as censors can easily block access to circumvention systems that make observable communication patterns. Moreover, the lack of unobservability may pose serious threats to their users. Recent research takes various approaches to tackle this problem, however they introduce new challenges, and the provided unobservability is breakable.**

In this paper we propose an easy-to-deploy and unobservable censorship-resistant infrastructure, called FreeWave. FreeWave works by modulating a client's Internet traffic into acoustic signals that are carried over VoIP connections. Such VoIP connections are targeted to a server, the FreeWave server, that extracts the tunneled traffic and proxies them to the uncensored Internet. The use of actual VoIP connections, as opposed to traffic morphing, allows FreeWave to relay its VoIP connections through oblivious VoIP nodes (e.g., Skype supernodes), hence keeping the FreeWave server(s) unobservable and unblockable. In addition, the use of end-to-end encryption, which is supported/mandated by most VoIP providers like Skype, prevents censors from distinguishing FreeWave's VoIP connections from regular VoIP connections.

To utilize a VoIP connection's throughput efficiently we design communications encoders tailored specifically for VoIP's lossy channel. We prototype FreeWave over Skype, the most popular VoIP system. We show that FreeWave is able to reliably achieve communication throughputs that are sufficient for web browsing, even when clients are far distanced from the FreeWave server. We also validate FreeWave's communication unobservability against traffic analysis and standard censorship techniques.

***Keywords*-Internet Censorship; Circumvention; VoIP;**

I. Introduction

The Internet is playing an ever-increasing role in connecting people from across the world, facilitating the free circulation of speech, ideas and information. This poses serious threats to repressive regimes as it elevates their citizens' awareness and provides them a powerful medium to arrange coordinated opposition movements. The recent unrest in the Middle East [1] demonstrates the very strong power of the Internet in arranging nation-wide protests that, in several cases, resulted in revolutionizing or even overthrowing repressive regimes. In response to such threats, repressive regimes make use of different technologies to restrict and monitor their citizens' access to the Internet; i.e., they *censor* the Internet. Censorship devices leverage various techniques [2], [3] ranging from simple IP address blocking and DNS hijacking to the more complicated and resource-intensive deep packet inspection (DPI) in order to enforce their blocking and monitoring. Citizens identified as non-complying with the censors' restrictions can face different consequences ranging from Internet service disruption to severe life-threatening punishments [4].

To help censored users gain open access to the Internet different systems and technologies have been designed and developed [5]–[11], generally referred to as *censorship circumvention* tools. These systems are composed of computer and networking technologies that allow Internet users to evade monitoring, blocking, and tracing of their activities. We observe that *the biggest challenge facing the existing circumvention systems is the lack of "unobservability"*: while these systems can, under certain conditions, circumvent censorship they are not effectively able to hide the fact that their users are making use of them [5]–[9]. For instance, the Tor [8] anonymity network is not able to effectively evade censorship as a censor can block all of the publicly advertised IP addresses of Tor relays. This has two major consequences: first, users caught (by censors) leveraging these circumvention systems may face various punishments such as imprisoning. Second, and even more catastrophic, this lack of unobservability usually leads to the lack of *availability*; i.e., circumvention systems with observable communication are easily blocked by censors. Censors proactively [12] look for Internet services that help with censorship circumvention and either block any access to them by their citizens, or leave them (partially) open to identify their users. In particular, censors rigorously look for IP addresses belonging to circumvention technologies (e.g., HTTP/SOCKS proxies) and add them to the IP blacklists maintained by their censoring firewalls [2], [13]. Consequently, citizens under repressive regimes often find it difficult to access the existing circumvention systems. For instance, the popular Tor network has frequently been/is blocked by several repressive regimes [12], [14].

To provide unobservable circumvention different approaches have been taken by the research community. Several systems [5], [7], [15] provide unobservability by *pre-sharing secrets* with their intended clients. The Tor system, for instance, has recently deployed Tor bridges [15], which are volunteer proxies whose IP addresses are distributed among Tor users in a selective manner. This makes Tor bridges less prone to be identified by censors, as compared to the publicly-advertised Tor entry nodes, however there are serious challenges in distributing their IP addresses among users [16], [17]. In a similar manner, Infranet [5] and Collage [7] aim for unobservability by pre-sharing some secret information with their users. This, however, is neither scalable nor effective as it is challenging to share secrets with a large number of real users, while keeping them secret from censors at the same time [18]–[20].

As another approach to provide unobservability, several systems use various *obfuscation* techniques. For instance, Ultrasurf [21] and Psiphon [22] try to confuse content filtering tools by obfuscating their design and traffic patterns. Such obfuscation, however, jeopardizes users' security, as analyzed in a recent study [23]. Appelbaum et al. propose *pluggable transports* [24] for Tor, a platform that allows one to build protocol-level obfuscation plugins for Tor traffic. These plugins obfuscate a Tor client's traffic to Tor bridges by shaping it to look like another protocol that is allowed by censors. Obfsproxy [25] is the first Tor pluggable transport. It adds an additional layer of encryption to Tor traffic to obfuscate Tor's content identifiers, like the TLS parameters; however, it does not remove Tor's statistical patterns like packet timings and sizes. Murdoch et al. [26] mention several weaknesses for obfsproxy, including being susceptible to either an active or passive attacker who has recorded the initial key exchange. StegoTorus [27] provides better unblockability, but comes with a much higher overhead [26]. SkypeMorph [28] morphs Tor traffic into Skype video calls in order to make it undetectable against deep-packet inspection and statistical analysis. The common issue with the aforementioned traffic obfuscation techniques is that they only obfuscate communication patterns, but not the end-hosts. In other words, while a censor may find it hard to detect the obfuscated traffic using traffic analysis, it will be able to identify the end-hosts that obfuscate the traffic through other active/passive attacks, e.g., SkypeMorph and StegoTorus relays can be enumerated using prevalent port knocking techniques [12], [29], zig-zag [30] attack, and insider attack [31]. Once the identity of a circumventing end-host is known to a censor the unobservability is completely lost and the end-host is easily blocked by the censor. CensorSpoofer [31] is another recent proposal that performs traffic obfuscation by mimicking VoIP traffic. Like most of the other designs noted above, CensorSpoofer needs to pre-share some secret information with the clients, posing a scalability challenge. In addition, it requires a usable upstream channel for its operation since its circumvented traffic is unidirectional.

As another recent trend, several proposals have sought unobservability by integrating circumvention into the Internet infrastructure [10], [11]. For instance Telex [10] and Cirripede [11] conceal the circumvented traffic inside the regular HTTPS traffic thanks to friendly ISPs that deflect/manipulate the intercepted connections. The real-world deployment of such circumvention systems requires collaboration of several trusted ISPs that make software and/or hardware modifications to their infrastructure; this does not seem to be realized in short-time until there are enough financial/political motives for the ISPs. Moreover, a recent study [32] shows that an adversary capable of changing routing decisions is able to block these systems.

In this paper we propose *FreeWave*, a censorship circumvention infrastructure that is highly unobservable (hence, highly available). The main idea of FreeWave, as shown in Figure 1, is to tunnel Internet traffic inside non-blocked VoIP communications by modulating them into acoustic signals that are carried over VoIP connections. For a censored user to use FreeWave for circumvention, she needs to setup a VoIP account with a public VoIP provider, and also to install FreeWave's client software on her machine. Part of the FreeWave system is a FreeWave server that listens on several *public*ly advertised VoIP IDs to serve FreeWave clients. To make a FreeWave connection, a user's FreeWave client software makes VoIP connections to FreeWave server's VoIP IDs. The client and server, then, tunnel the circumvented Internet traffic inside the established VoIP connections, by modulating network packets into acoustic signals carried by the established VoIP connections.

We claim that FreeWave provides strong unobservability by performing two kinds of obfuscations: *traffic obfuscation*, and *server obfuscation*. First, as FreeWave tunnels Internet traffic inside *actual*, encrypted VoIP connections, its traffic patterns are very hard to be distinguished from benign VoIP connections. Traffic obfuscation is also aimed for by recent morphing-based techniques like SkypeMorph [28] and StegoTorus [27], however, FreeWave provides stronger traffic obfuscation as it completely runs the target protocol instead of partially imitating it. The second obfuscation performed by FreeWave, which is *unique* to FreeWave, is *server obfuscation*, which prevents censors from detecting circumvented traffic by matching the destination addresses of traffic. Server obfuscation is an important feature that similar circumvention systems such as SkypeMorph [28] and StegoTorus [27] fail to provide. As we describe later in this paper, the way the FreeWave server is connected to the Internet results in getting FreeWave's VoIP traffic relayed by various, *oblivious* VoIP peers, preventing a censor from blocking/identifying FreeWave's VoIP traffic based on IP addresses (see Figure 1). For instance, FreeWave connections made through Skype get relayed by Skype *supern-*

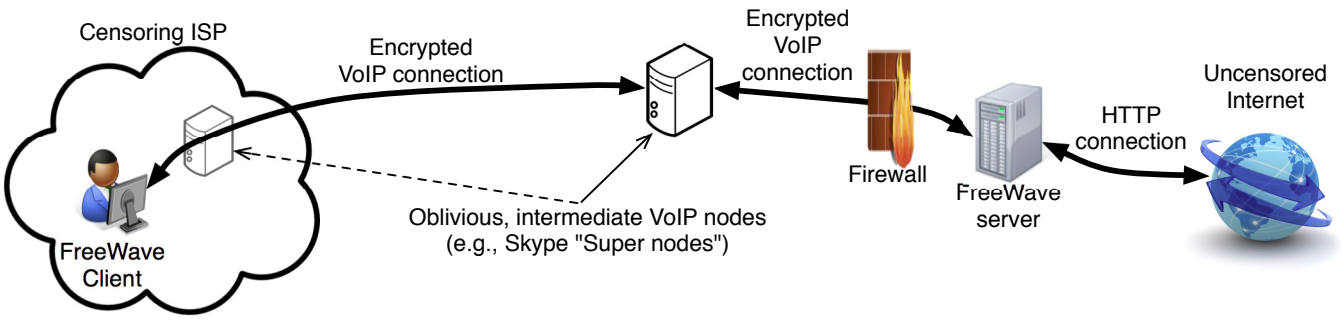

Figure 1. The main architecture of FreeWave.

odes [33], which are oblivious Skype users residing *outside*[1] *the censorship region*. As another example, if FreeWave uses Google Voice, FreeWave connections will get relayed by Google servers that are oblivious to the circumvention process. Server obfuscation, as defined above, is missing in *all* previous designs except CensorSpoofer [31]. For instance, in the case of Tor pluggable transports like SkypeMorph [28] and StegoTorus [27] once the IP address of the deploying Tor bridge is revealed to a censor (e.g., using port knocking [12], [16], [17], [29]) the unobservability is lost and the censor will be able to identify/block users connecting to that Tor bridge. In FreeWave, on the other hand, *even if a censor identifies the IP address belonging to a FreeWave server it will not be able to block connections to it* since users' connection to that FreeWave server are not direct connections, but are relayed through varying, oblivious VoIP nodes. We provide a thorough comparison of FreeWave with similar obfuscation-based techniques in Section IX.

The strong unobservability of FreeWave makes it highly unblockable (i.e., available). FreeWave's availability is tied to the availability of the VoIP service: since *the operation of FreeWave is not bound to a specific VoIP provider*, in order to block FreeWave a censor needs to block *all* VoIP connections with the outside world. This is not desirable by the censoring ISPs due to different business and political implications. VoIP constitutes an important part of today's Internet communications [36]–[38]; a recent report [37] shows that about one-third of U.S. businesses use VoIP solutions to reduce their telecommunications expenses, and the report predicts the VoIP penetration to reach 79% by 2013, a 50% increase compared to 2009.

We implement a prototype of FreeWave over the popular VoIP service of Skype and measure its performance. To achieve reliable communication over VoIP connections we design a communication encoder/decoder tailored for the VoIP's lossy communication channel. Specifically, we take advantage of Turbo codes and QAM modulation techniques [39], [40] in order to reliably encode the circumvented traffic inside the VoIP connections. Our evaluations show that FreeWave provides connection bit rates that are suitable for regular web browsing. We validate FreeWave's usability by clients that are geographically far away from the FreeWave server.

Contributions: In this paper we make the following main contributions:

- i) We propose, FreeWave, a novel infrastructure for censorship circumvention that works by modulating Internet traffic into the acoustic signals carried over VoIP connections. The use of actual VoIP connections, as well as being relayed by oblivious VoIP nodes provides promising unobservability for FreeWave.
- ii) We design communication encoders and decoders to efficiently modulate Internet traffic into acoustic signals.
- iii) We prototype FreeWave on the popular VoIP service of Skype and evaluate its performance and security.

The rest of this paper is organized as follows: In Section II we review our threat model and the goals in designing our circumvention system. We describe the design of our proposed circumvention system, FreeWave, in Section III, and Section IV discusses our design details. In Section V, we discuss the features of our designed circumvention system. We thoroughly analyze the security of FreeWave in Section VI. In Section VII we describe the design of MoDem, the communication block of FreeWave software. We describe our prototype implementation in Section VIII along with the evaluation results. In Section IX we compare FreeWave with two recent proposals of SkypeMorph [28] and CensorSpoofer [31]; this is followed by additional related work in Section X. In Section XI we discuss FreeWave's limitations and several recommendations. Finally, the paper is concluded in Section XII.

II. Preliminaries

A. Threat model

We assume that a FreeWave client is connected to the Internet through a censoring ISP, e.g., an ISP that is

[1]The supernodes assigned to a particular Skype client by the Skype protocol are geographically close to that client for better quality of service; hence a FreeWave server is expected to use nearby supernodes. In addition, a FreeWave server can adjust the list of its Skype supernodes [34], [35], as described later.

controlled and regulated by a repressive regime. Based on the regulations of the censoring ISP its users are not allowed to connect to certain Internet destinations, called the *censored destinations*. The users are also prohibited from using censorship circumvention technologies that would help them to evade the censoring regulations. The censoring ISP uses a set of advanced technologies to enforce its censoring regulations, including IP address blocking, DNS hijacking, and deep packet inspection [2], [3]. The censoring ISP also monitors its users' network traffic to identify and block any usage of censorship circumvention tools; traffic analysis can be used by the censor as a powerful technique for this purpose.

We assume that the censoring ISP enforces its regulations such that it does not compromise the *usability* of the Internet for its users, due to different political and economical reasons. In other words, the enforced censorship does not disable/disrupt key Internet services. In particular, we consider VoIP as a key Internet service in today's Internet [36], [38], [41], and we assume that, even though a censor may block certain VoIP providers, the censor will not block *all* VoIP services. VoIP constitutes a key part in the design of FreeWave.

B. Design goals

We consider the following goals in the design and evaluation of FreeWave. Later in Section V, we discuss these features for the FreeWave circumvention system proposed in this paper and compare FreeWave with related work.

Unblockability: The main goal of a censorship circumvention system is to help censored users gain access to censored Internet destinations. As a result, the most trivial property of a circumvention system is being accessible by censored users, i.e., it should be unblockable by censors.

Unobservability: Unobservability is to hide users' utilization of a circumvention system from censorship authorities, which is a challenging feature to achieve due to the recent advances in censorship technologies [2]. The importance of unobservability is two-fold; first, an observable circumvention can jeopardize the safety of a user who has been caught by the censor while using the circumvention system. Second, a weak unobservability commonly results in a weak unblockability, as it allows censors to more easily identify, hence block, traffic generated by the circumvention system.

Security: Several security considerations should be made once analyzing a circumvention system. These considerations include users' anonymity, confidentiality, and privacy against various parties including the censors, the circumvention system, and third parties.

Deployment feasibility: An important feature of a circumvention system is the amount of resources (e.g., hardware, network bandwidth, etc.) required for it to be deployed in real world. A circumvention system is also desired to have few dependencies on other systems and entities in order to make it more reliable, secure, and cost-effective.

Quality of service: A key feature in making a circumvention system popular in practice is the quality of service provided by it in establishing circumvented connections. Two important factors are connection bandwidth, and browsing latency.

III. FreeWave scheme

In this section, we describe the design of FreeWave censorship circumvention. Figure 1 shows the main architecture of FreeWave. In order to get connected through FreeWave, a user installs a *FreeWave client* on her machine, which can be obtained from an out-of-band channel, similar to other circumvention systems. The user sets up the installed FreeWave client by entering her own VoIP ID and also the publicly advertised VoIP ID of FreeWave server. Once the FreeWave client starts up, it makes a VoIP audio/video call to FreeWave server's VoIP ID. As discussed in Section IV-B, the FreeWave server is configured in a way that VoIP connections initiated by clients are relayed through various *oblivious VoIP peer*s, e.g., Skype supernodes; this is a key security feature of FreeWave as it prevents a censor from blocking FreeWave's VoIP connections using IP address blocking. Also, since FreeWave's VoIP connections are end-to-end encrypted, a censor will not be able to identify FreeWave's VoIP connections by analyzing traffic contents, e.g., by looking for the VoIP IDs. Using the established VoIP connection, a FreeWave client circumvents censorship by modulating its user's Internet traffic into acoustic signals that are carried over by such VoIP connections. FreeWave server demodulates a client's Internet traffic from the received acoustic signals, and proxies the demodulated traffic to the requested Internet destinations.

Next, we introduce the main components used in FreeWave and describe how these components are used in the design of FreeWave's client and server.

A. Components of FreeWave

In this section, we introduce the main elements used in the design of FreeWave client and server software. The first three components are used by both FreeWave client and FreeWave server, while the fourth element is only used by FreeWave server.

VoIP client VoIP client is a Voice-over-IP (VoIP) client software that allows VoIP users to connect to one (or more) specific VoIP service(s). In Section IV-B, we discuss the choices of the VoIP service being used by FreeWave.

Virtual sound card (VSC) A virtual sound card is a software application that uses a physical sound card installed on a machine to generate one (or more) isolated, virtual sound card interfaces on that machine. A virtual sound card interface can be used by any application running on the host

machine exactly the same way a physical sound card is utilized. Also, the audio captured or played by a virtual sound card does not interfere with that of other physical/virtual sound interfaces installed on the same machine. We use virtual sound cards in the design of FreeWave to isolate the audio signals generated by FreeWave from the audio belonging to other applications.

MoDem FreeWave client and server software use a modulator/demodulator (MoDem) application that translates network traffic into acoustic signals and vice versa. This allows FreeWave to tunnel the network traffic of its clients over VoIP connections by modulating them into acoustic signals. We provide a detailed description of our MoDem design in Section VII.

Proxy FreeWave server uses an ordinary network proxy application that proxies the network traffic of FreeWave clients, received over VoIP connections, to their final Internet destinations. Two popular choices for FreeWave's proxy are the HTTP proxy [42] and the SOCKS proxy [43]; a SOCKS proxy supports proxying of a wide range of IP protocols, while an HTTP proxy only supports proxying of HTTP/HTTPS traffic, but it can perform HTTP-layer optimizations like pre-fetching of web contents. Several proxy solutions support both protocols.

B. Client design

The FreeWave client software, installed by a FreeWave user, is consisted of three main components described above: a VoIP client application, a virtual sound card (VSC), and the MoDem software. Figure 2 shows the block diagram of the FreeWave client design. MoDem transforms the data of the network connections sent by the web browser into acoustic signals and sends them over to the VSC component. The FreeWave MoDem also listens on the VSC sound card to receive specially formatted acoustic signals that carry modulated Internet traffic; MoDem extracts the modulated Internet traffic from such acoustic signals and sends them to the web browser. In a sense, the client web browser uses the MoDem component as a network proxy, i.e., the listening port of MoDem is entered in the HTTP/SOCKS proxy settings of the browser.

The VSC sound card acts as a bridge between MoDem and the VoIP client component, i.e., it transfers audio signals between them. More specifically, the VoIP client is set up to use the VSC sound card as its "speaker" and "microphone" devices (VoIP applications allow a user to select physical/virtual sound cards). This allows MoDem and the VoIP client to exchange audio signals that contain the modulated network traffic, isolated from the audio generated/recorded by other applications on the client machine.

For the FreeWave client to connect to a particular FreeWave server it *only* needs to know the VoIP ID belonging to that FreeWave server, but not the IP address of the FreeWave server. Every time the user starts up the FreeWave client application on her machine the VoIP application of FreeWave client initiates an audio/video VoIP call to the known VoIP ID of the FreeWave server.

C. Server design

Figure 3 shows the design of FreeWave server, which consists of four main elements. FreeWave server uses a VoIP client application to communicate with its clients through VoIP connections. A FreeWave server chooses one or more VoIP IDs, which are provided to its clients, e.g., through public advertisement.

The VOIP client of the FreeWave server uses one (or more) virtual sound cards (VSC) as its "speaker" and "microphone" devices. The number of VSCs used by the server depends on the deployment scenario, as discussed in Section IV-A. The VSC(s) are also used by the MoDem component, which transforms network traffic into acoustic signals and vice versa. More specifically, MoDem extracts the Internet traffic modulated by FreeWave clients into audio signals from the incoming VoIP connections and forwards them to the last element of the FreeWave server, FreeWave *proxy*. MoDem also modulates the Internet traffic received from the proxy component into acoustic signals and sends them to the VoIP client software through the VSC interface. The FreeWave proxy is a regular network proxy, e.g., an HTTP proxy, that is used by the FreeWave server to connect FreeWave clients to the open Internet. As mentioned above in Section III-B, the web browser of a FreeWave client targets its traffic to a network proxy; such proxied traffic is received and handled by FreeWave server's proxy server (through the VoIP connections, as described).

IV. Other Design Details

A. Deployment scenarios

The FreeWave system proposed in this paper can be deployed by "good" entities that run FreeWave servers to help censored users gain an uncensored access to the Internet. We consider the following scenarios for a real-world deployment of FreeWave. In Section VI, we discuss the security considerations for each of these scenarios.

Personal deployment: A person having an open access to the Internet can set up a personal FreeWave server on her personal machine, *anonymously* helping censored users evade censorship. Such a person can, then, advertise her VoIP ID (used with her FreeWave server) publicly (e.g., through social networks) and anyone learning this ID would be able to connect to the Internet by running FreeWave client software. To save bandwidth, she can configure her FreeWave server to enforce restrictions on the quality of service provided to clients.

Central VoIP-center: FreeWave service can be deployed and maintained by a central authority, e.g., a for-profit or non-profit organization. The deploying organization can

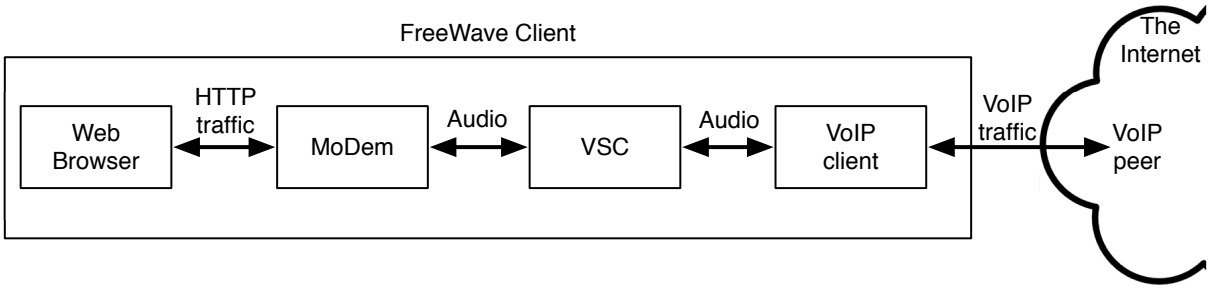

Figure 2. The main components of FreeWave client.

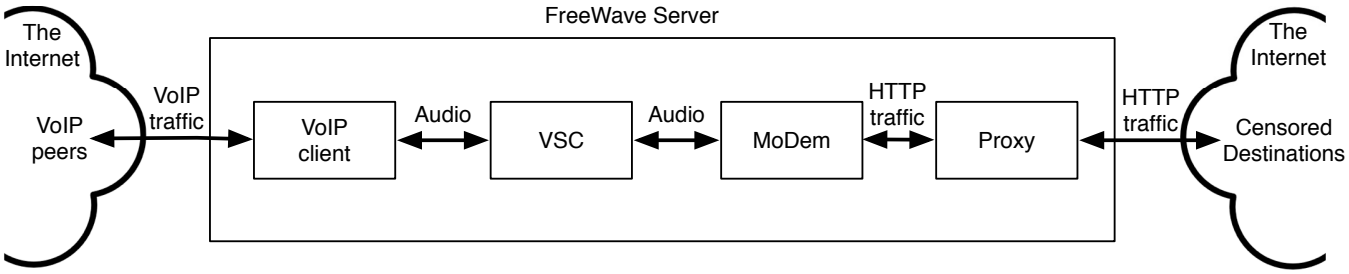

Figure 3. The main components of FreeWave server.

build and run FreeWave servers that are a capable of serving large numbers of FreeWave clients. To do so, the deployed FreeWave servers should utilize several physical/virtual sound cards in parallel. Also, by creating VoIP accounts on several, different VoIP service providers such central FreeWave system will be able to service FreeWave clients who use various VoIP services. Such a central deployment of FreeWave can operate for commercial profit, e.g., by charging clients for the used bandwidth, or can be established as a non-profit system, e.g., being funded by NGOs or pro-freedom governments.

Central phone-center: As an alternative approach, FreeWave can be deployed using an automated telephone center. More specifically, instead of VoIP IDs, FreeWave will publicize several phone numbers, which are used by clients to connect to the FreeWave server. FreeWave users need to use the exact same FreeWave client software, except that instead of making VoIP calls to a VoIP IDs they will make VoIP calls to FreeWave server's phone numbers. Compared to the "central VoIP-center" scenario, this has the big advantage that clients can arbitrarily choose any VoIP service provider for the client software, while in the "central service" design users need to choose from the VoIP systems supported by FreeWave server (though a powerful FreeWave server can support many VoIP systems).

Distributed service: FreeWave service can also be deployed in a distributed architecture, similar to that of Tor [8] anonymity network. More specifically, a FreeWave network can be built consisting of a number of volunteer computers that run instances of FreeWave server software on their machines. A central authority can manage the addition of new volunteer nodes to the system and also the advertisement (or distribution) of their VoIP IDs to the clients.

B. The choice of VoIP systems

There are numerous free/paid *VoIP service provider*s that can be utilized by the FreeWave system, e.g., Skype[2], Vonage[3], iCal[4], etc. A VoIP service provider usually supplies its VoIP client software to its users, but there are also some VoIP software that can be used for different VoIP accounts, e.g., PhonerLite[5]. In this section, we mention some candidate VoIP services that can be used by FreeWave.

1) Skype: Skype is a peer-to-peer VoIP system that provides voice calls, instant messaging, and video calls to its clients over the Internet. Skype is one of the most popular VoIP service providers with over 663 million users as of September 2011 [44].

Skype uses an undisclosed proprietary design, which has been partly reverse-engineered in some previous research [34], [35], [45]. These studies find that Skype uses a peer-to-peer overlay network with the Skype users as its peers. There are two types of nodes on Skype: *ordinary nodes*, and *supernodes (SN)*. Any Skype client with a public IP address, having sufficient CPU, memory, and network bandwidth serves as a supernode, and all the other nodes are ordinary nodes. In addition, Skype uses a central *login*

[2] http://www.skype.com
[3] http://www.vonage.com
[4] http://www.icall.com/
[5] http://www.phonerlite.de/index_en.htm

server that keeps users' login credentials and is used by Skype users to register into Skype's overlay network. Apart from the login server all Skype communications work in a peer-to-peer manner, including the user search queries and online/offline user information.

A key feature that makes Skype an ideal choice for FreeWave is its peer-to-peer network. Depending on its network setting [33], an ordinary Skype user deploys some supernodes as her proxies to connect to the Skype network, to make/receive calls, and to update her status. In particular, a Skype call made toward an ordinary Skype node gets relayed to her by her supernodes [34], [35]. Each ordinary node maintains a *supernode-cache* [35] table that keeps a list of reachable (usually nearby) supernodes, discovered by the Skype protocol. We use this feature to provide server obfuscation for FreeWave: by having our FreeWave server to act as an ordinary Skype node the VoIP connections that it receives will be relayed by alternative supernodes, rendering IP address blocking impossible. We discuss this further in Section VI. Also note that a censor can not map a FreeWave server to its supernodes since the supernode-cache table is a large, dynamic list; further, a Skype client can change its supernodes more frequently by *flush*ing [34], [35] its supernode-cache.

Based on the criteria mentioned for a supernode, an easy way to be treated as an ordinary node by Skype is to reside in a firewalled, NATed network subnet [33], [35]. As another interesting feature of Skype for FreeWave, all Skype connections are secured by end-to-end encryption [34], [35].

2) SIP-based VoIP: Session Initiation Protocol (SIP) [46] is a light-weight, popular signaling protocol and is widely used by VoIP providers, e.g., SFLphone[6], Zfone[7], and Blink[8], to establish calls between clients. A SIP-based VoIP system consists of three main elements [46]: 1) *user agents* that try to establish SIP connections on behalf of users, 2) a *location service* that is a database keeping information about the users, and 3) a number of *servers* that help users in establishing SIP connections. In particular, there are two types of SIP servers; *registrar* servers receive registration requests sent by user agents and update the location service database. The second types of SIP servers are *proxy* servers that receive SIP requests from user agents and other SIP proxies and help in establishing the SIP connections.

Once a SIP connection is established between two user agents a media delivery protocol is used to transfer media between the users. Most of the SIP-based VoIP systems use the Real-time Transport Protocol (RTP) [47] to exchange audio data, and the Real-Time Transport Control Protocol (RTCP) [47] protocol to control the established RTP connections. User agents in SIP-based VoIP system are allowed to use an encryption-enabled version of RTP, called Secure Real-time Transport Protocol (SRTP) [48], in order to secure their VoIP calls. Note that the encryption supported by SRTP is performed end-to-end by SIP agents and VoIP servers are not required to support encryption. We mandate the SIP-based design of FreeWave to use SRTP for media transfer.

Similar to Skype, if a user agent is behind NAT or a firewall it will use an intermediate node to establish its VoIP connections. In particular, two popular techniques used by VoIP service providers to bypass NAT and firewalls are *session border controller* (SBC) [49] and *RTP bridge server*s [50]. As in the case of the Skype-based FreeWave, putting a FreeWave server behind a firewall masks its IP address from censors, as the VoIP calls to it will be relayed through oblivious intermediate nodes. However, better care needs to be taken in this case since, unlike Skype, SIP-based VoIP systems are not peer-to-peer.

3) Centralized VoIP: Several VoIP providers use their own servers to relay VoIP connections, in order to improve connectivity, regardless of the VoIP protocol that they use. One interesting example is the Google Voice[9], which relays all of its calls through Google servers, hence disguising a callee's IP address from a censor. Also note that the calls in Google Voice are encrypted.

V. Evalution of the design goals

In Section II-B, we listed several features that we consider in designing an effective circumvention system. Here, we discuss the extent to which our proposed system, FreeWave, achieves such requirements.

Unblockability: In order to use FreeWave, a client only needs to know the VoIP ID of the FreeWave server, i.e., `server-id`, but no other secret/public information like the server's IP address. `server-id` is distributed in a public manner to the users, so we assume that it is also known to censors. Considering the use of encrypted VoIP connections by FreeWave, this public knowledge of `server-id` does not allow censors to identify (and block) the VoIP connections to the FreeWave server. In addition, a censor will not be able to identify FreeWave's VoIP connections from their IP addresses since, as discussed in Section IV-B, the encrypted VoIP connections to the FreeWave server are relayed through oblivious, intermediate nodes (given the FreeWave server is set up appropriately). For instance, in Skype-based FreeWave the VoIP connections to the FreeWave server are relayed by oblivious Skype supernodes. Also, FreeWave server is not mapped to a particular set of supernodes, i.e., its VoIP connections are relayed through a varying set of super nodes. In all of the above arguments, we assume that the VoIP service provider used by FreeWave is not colluding with the censors, otherwise the unobservability is lost. Such collusion could happen if a centralized VoIP

[6] http://sflphone.org/
[7] http://zfoneproject.com/
[8] http://icanblink.com/
[9] https://www.google.com/voice

service, e.g., Google Voice, informs censors of the clients calling FreeWave's Google Voice ID, or if the censors control the supernodes used by a FreeWave server.

Another point in making FreeWave unblockable is that it does not depend on a particular VoIP system, and can select from a wide range of VoIP providers. As a result, in order to block FreeWave, censors will need to block *all* VoIP services, which is very unlikely due to several political and economical considerations.

Note that unblockability is a serious challenge with many existing circumvention systems, as the very same information that they advertise for their connectivity can be used by censors to block them. For example, the Tor [8] system requires its clients to connect to a public set of IP addresses, which can be IP-filtered by censors. More recently, Tor has adopted the use of Tor *bridges* [15], which are volunteer proxies with semi-public IP addresses. Unfortunately, there are different challenges [12], [16], [17], [20], [29] in distributing the IP addresses of Tor bridges only to real clients, but not to the censors.

Unobservability: The arguments made above for FreeWave's unblockability can also be used to justify its unobservability. As mentioned above, even though FreeWave server's VoIP ID (`server-id`) is assumed to be known to censors, the end-to-end encryption of VoIP connections prevents a censor from observing users making VoIP connections to `server-id`. In addition, VoIP relays sitting between FreeWave clients and a FreeWave server, e.g., Skype supernodes, foil the identification of FreeWave connections through IP address filtering.

Deployment feasibility: The real-world deployment of FreeWave does not rely on other entities. This is in contrast to some recent designs that need collaboration from third parties for their operation. For instance, Infranet [5] requires support from some web destinations that host the circumvention servers. As another example, several recent proposals [10], [11], [51] rely on the collaboration from friendly ISPs for their operation.

Quality of service: In Section VIII, we discus the connection performance provided by our prototype implementation of FreeWave. Our results show that FreeWave provides reliable connections that are good for normal web browsing.

VI. Security analysis

In this section, we discuss the security of FreeWave clients to the threats imposed by different entities.

A. Security against censors

The end-to-end encryption of VoIP connections protects the confidentiality of the data sent by FreeWave clients against a monitoring censor, even if the censor is able to identify VoIP connections targeted to FreeWave. Such end-to-end encryption also ensures the web browsing privacy of FreeWave clients. As mentioned in Section IV-B, Skype calls are encrypted end-to-end, and SIP-based VoIPs also provide end-to-end encryption using the SRTP protocol. In the case of centralized VoIP services, like the Google Voice, the encryptions are usually client-to-server, hence the FreeWave client should ensure that its VoIP provider is not colluding with the censors.

Even though FreeWave uses encrypted VoIP connections a censor may still try to identify FreeWave-generated VoIP connections by performing traffic analysis, i.e., by analyzing communication patterns. The use of actual VoIP connections by FreeWave (instead of shaped connections as in [27], [28]) makes traffic analysis particularly hard. We show this in Section VIII-C by analyzing FreeWave's VoIP connections and comparing them with regular VoIP connections. As discussed in Section VIII-C, the choice of the VoIP system affects the feasibility of traffic analysis. Please see Section VIII-C for more discussion on FreeWave traffic analysis.

B. Security against FreeWave servers

A FreeWave server only knows the VoIP IDs of its client, but not their IP addresses since the VoIP connections are being relayed through intermediate VoIP nodes. As a result, unless the VoIP service (e.g., the Google Voice server, or a Skype supernode owned by a FreeWave server) is colluding with the FreeWave server, the FreeWave server will not be able to link VoIP IDs to IP addresses, i.e., the client is anonymous to the server. Note that anonymity against circumvention systems is not demanded by typical censored users who are only willing to access non-sensitive censored information like the news, and in fact some popular circumventions mechanisms do not provide such anonymity, e.g., the single-proxy based systems such as the Anonymizer [9]. A FreeWave client can strengthen its anonymity against the FreeWave server in different ways. For instance, she can enforce its VoIP traffic to be relayed by additional intermediate VoIP relays, e.g., by the client's Skype supernodes.

In the basic design of FreeWave mentioned above a FreeWave server can observe the traffic contents exchanged by a FreeWave client, since the tunneled traffic is not always encrypted. However, a client can easily ensure security and privacy from the server by using an extra layer of encryption. For instance, a client can use FreeWave to get connected to an anonymity system like Anonymizer [9], and then use the tunneled connection with this anonymity system to browse the Internet. This secures this client's traffic from the FreeWave server, as well as makes it confidential. Note that considering the fact that FreeWave clients are anonymous to FreeWave servers, clients may opt not to use such an additional protection for low-sensitive activities like web browsing.

C. Security against VoIP providers

Except for the centralized VoIP services, the VoIP connections between FreeWave clients and servers are encrypted end-to-end using the keys shared through the VoIP protocol. In the case of a centralized VoIP service, like the Google Voice, FreeWave parties can exchange a key using a key sharing mechanism, like the Diffie-Hellman key exchange [52], over the established FreeWave VoIP. As a result, the VoIP provider will not be able to observe the data being communicated, nor the web destinations being browsed. However, the VoIP service provider might be able to identify VoIP IDs that have made VoIP calls to a FreeWave server. As a result, in order to ensure its unobservability FreeWave needs to use VoIP providers that are not colluding with the censors. Note that FreeWave does not rely on a particular VoIP system and any VoIP provider can be used for its operation.

VII. FreeWave MoDem

The MoDem component is one of the main components of both FreeWave client and FreeWave server application, which translates Internet traffic into acoustic signals and vice versa. MoDem consists of a *modulator* and a *demodulator*. MoDem's modulator modulates data (IP bits) into acoustic signals, and MoDem's demodulator extracts the encoded data from a received acoustic signal. In the following, we describe the design of MoDem's modulator and demodulator.

A. Modulator description

We design a *bit-interleaved coded modulation* (BICM) [40] for MoDem's modulator, which is shown in Figure 4. First, the modulator encodes the information bits, $\{a_i\}$, i.e., IP traffic, using a channel encoder with rate R_c. The encoded stream, $\{b_i\}$, is permuted using a random interleaver [40], and the interleaved sequence is, then, partitioned into subsequences $\boldsymbol{c}_n = \{c_n^1, \ldots, c_n^Q\}$ of length-Q (n is the partition index and Q is a parameter of our modulator). Finally, a QAM mapper [40] generates the modulated data by mapping each subsequence $\boldsymbol{c_n}$ to a 2^Q-ary quadrature amplitude modulation symbol.

We design a wrapper protocol to carry the modulated data. This wrapper performs three important tasks: 1) it allows a demodulator to synchronize itself with the modulator in order to correctly identify the starting points of the received data; 2) it lets the sender and receiver negotiate the modulation parameters; and, 3) it lets the demodulator adapt itself to the time-varying channel. Figure 4 shows the modulated data being wrapped by our wrapper protocol. As can be seen, the modulated bit stream is converted into data *frame*s that are sent over the VoIP channel. Each data frame starts with a known *preamble* block, which is needed for synchronization as well as for receiver initialization purposes. The frame preamble is followed by a *signal* block that is used to communicate the modulation and coding parameters used for this particular frame. The signal block is followed by N blocks of training and data symbols. The data symbols are the output of the QAM modulator. The training blocks are needed to adapt the demodulator to the time-varying channel.

The data frames, as generated above, are sent over the VoIP channel using acoustic signals. In particular, for x_n being the n-th symbol in a frame, the frame is mapped to a waveform $x(t) : \mathbb{R} \rightarrow \mathbb{C}$ as follows:

$$x(t) = \sum_l x_l p(t - lT) \tag{1}$$

where $p(t)$ is a basic pulse shifted by multiples of the symbol period T. This signal is then transformed to a passband [39] signal with the center frequency of f_C:

$$x_{PB}(t) = 2\Re\{x(t)e^{2\pi i f_C t}\} \tag{2}$$

which is then sent over the VoIP channel (by getting sent to the virtual sound card). $\Re\{\}$ returns the real component of a complex number, and i is the imaginary unit.

B. Demodulator description

Figure 5 shows MoDem's demodulator, which is designed to effectively extract the data that the modulator embedded into an audio signal. For an audio waveform, $r(t)$, received from the virtual sound card the demodulator shifts its spectrum by the center frequency f_C, passes it through a low-pass filter and then samples the resulting signal at symbol rate (equal to $1/T$). The synchronizer correlates the preamble block with the obtained samples, declares the point of maximum correlation as the starting point of the received frame, discards all samples before this point, and enumerates the remaining samples by $r_n (n = 1, 2, ...)$. We assume the voice channel to be linear and can hence write: [39]:

$$r_n = \sum_{k=-K_f}^{K_p} h_{n,k} x_{n-k} + w_n \tag{3}$$

where n and k are time and delay indices, respectively. Also, w_n is a complex white Gaussian noise process, which models the noise added to the modulated data as a result of the noisy channel (e.g., due to VoIP codec's lossy compression). Moreover, $h_{n,k}$ is the channel gain [39], which may vary in time. The channel length is assumed to be at most $K_f + K_p + 1$, where K_f is the length of the precursor and K_p is the length of the postcursor response.

The demodulator passes the discrete stream of $\{r\}$ through a Turbo equalizer [39]. The goal of this equalizer is to obtain an estimation of $\{x\}$, i.e., the discrete modulated data. The estimated data is passed to a channel decoder, which is the equivalent decoder for the encoder used by MoDem's modulator. We also put an interleaver and a

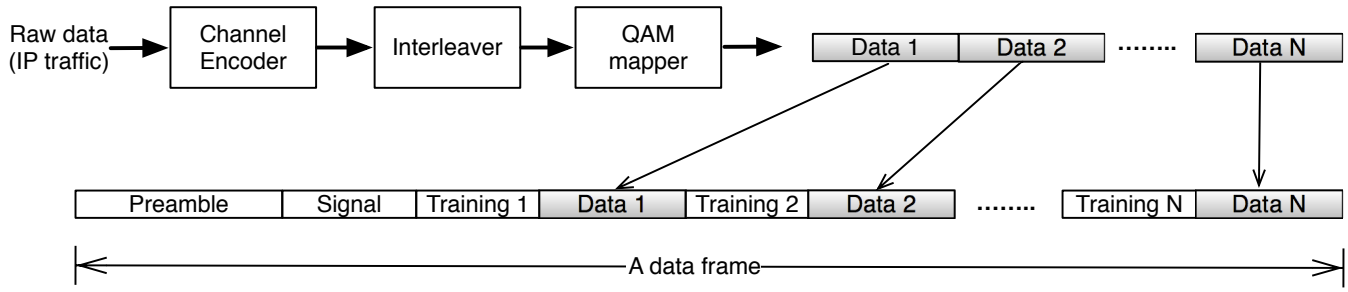

Figure 4. The modulator block of FreeWave's MoDem. The modulated data is wrapped by a wrapper protocol before being transformed into acoustic waveforms.

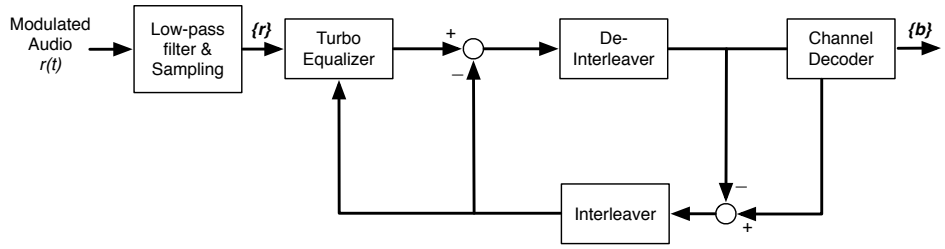

Figure 5. Block diagram of MoDem's demodulator.

de-interleaver block between the Turbo equalizer and the channel decoder modules; this is to uniformly distribute burst bit errors, generated in the channel, across the stream in order to improve the decoding process. This is because our channel decoder performs well with distributed errors, but poorly with bursty errors.

VIII. Prototype and Evaluation

In this section, we describe our prototype implementation and discuss its connection performance.

A. Implementation setup

We have built a prototype implementation of FreeWave over Skype. Our MoDem component uses Matlab's libraries for acoustic signal processing, and we use Virtual Audio Card [10] as our virtual sound card (VSC) software. We also use the free version of Skype client software[11] provided by Skype Inc. as our VoIP client component. Our MoDem software, as well as the Skype client are set up to use the Virtual Audio Card as their audio interface. We have built our FreeWave client and FreeWave server using the components mentioned above. In order to emulate a real-world experience, i.e., a long distance between a FreeWave client and a FreeWave server, we connect our FreeWave client to the Internet though a VPN connection. In particular, we use the SecurityKISS[12] VPN solution that allows us to pick VPN servers located in different geographical locations around the world. Note that this identifies the location of our FreeWave clients; our FreeWave server is located in Champaign, IL, USA.

MoDem specifications: Our evaluations show that the data rates that can be achieved with our system clearly depend on the bandwidth of the Internet connection and the distance between the client and server. The minimum bandwidth required for a voice call is 6 kbps for both upload and download speeds, according to Skype. For the pulse function of MoDem's modulator, $p(t)$ (Section VII), we use a square-root raised cosine filter with a roll-off factor 0.2 and a bandwidth of $1/T$. The carrier frequency f_C is chosen such that the spectrum of the voiceband is always covered. At the demodulator, the same square-root raised cosine filter is used for low-pass filtering. Our communication system automatically adjusts the symbol constellation size Q, the channel coding rate R_c, and the symbol period T such that the best possible data rate is achieved. The receiver knows how well the training symbols were received, and based on this feedback the modulator can optimize the data rate. The relationship between the data rate R and the above parameters is $R = (QR_c)/T$. Our designed demodulator is iterative [39]. The number of iterations needed for convergence depends on the channel condition, which is typically measured by means of the signal to noise power ratio, the SNR.

B. Connection performance

Connection data rates: Table I shows the bit rates achieved by FreeWave clients connecting from different geographic

[10] http://software.muzychenko.net/eng/vac.htm
[11] http://www.skype.com/intl/en-us/get-skype/
[12] http://www.securitykiss.com/

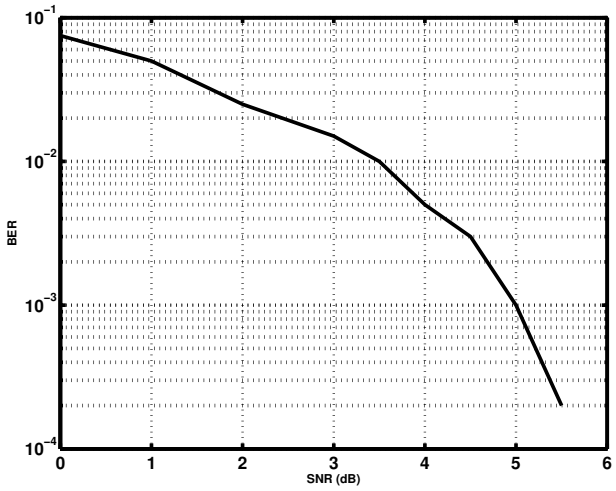

Figure 6. BER versus SNR for FreeWave.

locations to our FreeWave server, located in Champaign, IL, USA. At the beginning of each FreeWave connection, our client runs an assessment subprotocol to identify the best codecs and the reliable data rate. The table lists the best compromise between data rates and packet drop rates, for different clients. As can be seen, clients in different parts of Europe are reliably able to get connection bit rates of 16kbps by using FreeWave over Skype. Users within the US are able to achieve higher data rates, e.g., 19.2kbps for a client in Chicago, IL. Note that the distance between a FreeWave client and the FreeWave server slightly affects the achievable data rates. To illustrate this, Figure 6 shows the bit error rate (BER) performance of our designed demodulator for different SNRs in the log-scale for a 19kbps FreeWave connection. As can be seen, for SNRs larger than 5.4dB the BER tends to zero (the zero value cannot be shown in the log-scale figure). A distributed deployment of FreeWave can provide users from many different geographic locations with the same reliable data rate speeds; for instance, FreeWave servers running in Europe can assist FreeWave users from the Middle East better than the FreeWave servers that are located in the US.

Maximum achievable data rates: As illustrated above, our FreeWave prototype is able to reliably achieve bit rates of up to 19kbps, using the MoDem component designed in this paper. It is possible to design more complicated MoDems that can achieve higher bit rates, however, a MoDem will not be able to achieve arbitrarily large data rates. This is due to the fact that each VoIP codec samples speech at a particular rate (or at a given range of rates) [53] and FreeWave cannot achieve data rates higher than a codec's bit-rate. For instance, Skype generates a bit-rate between 6 and 40kbps [53] (depending on the distance between the end-hosts, Internet bandwidth and few other factors), resulting in a "maximum" achievable rate of 40kbps for FreeWave (the actual rate achieved depends on the efficiency of MoDem). The "L16" codec generates a 128kbps data rate, resulting in a maximum FreeWave bit-rate of up to 128kbps. As another instance, the widely used codec of "G.711" produces a 64kbps data rate [53], leading to a maximum FreeWave bit rate of 64kbps.

We believe that the bit rates achievable by the current design of FreeWave are enough for normal web browsing, especially for a user under a repressive regime who aims to do normal web browsing. On the other hand, a trivial approach to achieve much higher rates is to encode Internet traffic into the *video* signals carried over VoIP connections. This requires designing efficient modulator/demodulators for encoding data into video, which we leave for future research.

C. Traffic analysis

In order to resist traffic analysis, FreeWave VoIP connections should have communication patterns similar to that of regular VoIP connections. Note that FreeWave uses encrypted VoIP connections, so a censor will not be able to analyze packet contents (popular VoIP providers like Skype provide/mandate encrypted VoIP connections). The two traffic patterns that may be used for traffic analysis in this case are *packet rates* and *packet sizes*. Most of the standard VoIP codecs, like the widely used G.7 series [53], use fixed bit rates and fixed packet sizes during a given connections, or even across all connections [53]. This prevents any kind of traffic analysis against FreeWave connections that use these codecs. In fact, these codecs are widely used by different VoIP providers, e.g., the Google Voice service [54]. On the other hand, several VoIP codecs use variable bit-rates, most notably Skype's proprietary SILK [55] codec. When FreeWave uses a VoIP service that uses variable-bit-rate codecs, special care needs to be taken to prevent traffic analysis. We have analyzed the FreeWave traffic sent over Skype in our prototype implementation, and have compared its traffic patterns with regular Skype traffic. We observe that there are two states in a regular Skype call: "Skype-Speak", in which the callee is speaking over Skype, and "Skype-Silence", in which the callee is silent (e.g., she is listening to the person on the other side of the line).

Table II shows the average communication statistics for the three different types of Skype traffic, i.e, Skype in the Skype-Speak state, Skype in the Skype-Silent, and Skype tunneling FreeWave. All the analysis is done for the same pair of Skype peers. As can be seen from the table, FreeWave over Skype generates communication patterns very similar to regular Skype in the Skype-Speak state, while the Skype-Silent state generate lower packet rates and smaller packet sizes. This is because Skype's SILK [55] codec reduces its packet rate and uses smaller packets when the audio signal captured by Skype client is weak, in order to conserve bandwidth. We observe that, based on this analysis, a FreeWave over Skype call makes communication patterns very similar

Client location	MoDem parameters			Data rate	Packet drop rate
	Q	$1/T$	R_C		
Berlin, Germany	4	8 kHz	0.5	16000 bps	0
Frankfurt, Germany	4	8 kHz	0.5	16000 bps	0
Paris, France	4	8 kHz	0.5	16000 bps	0
Maidenhead, UK	4	8 kHz	0.5	16000 bps	0
Manchester, UK	4	8 kHz	0.5	16000 bps	0
Lodz, Poland	4	8 kHz	0.5	16000 bps	0.06
Chicago, IL	4	9.6 kHz	0.5	19200 bps	0.01
San Diego, CA	4	9.6 kHz	0.469	18000 bps	0

Table I
EVALUATION RESULTS OF FREEWAVE.

to a typical Skype call: In a typical Skype call, when one side of the connection is in the Skype-Speak state the other side is usually in the Skype-Silent state (i.e., listening to the other side). In a FreeWave over Skype call, also, when one side of the connection is sending data the other side is usually idle, e.g., a web traffic is a serious of HTTP GET and HTTP RESPONSE messages that appear in a sequence. Furthermore, simple modifications can be made to FreeWave client and server software in order to better hide its traffic pattern; for instance, one side can stop sending data if the other side is sending data, or a dummy audio can be sent if both sides have been silent for a long time. Once again, note that this is only required if FreeWave is deployed on a VoIP system that uses a variable-length audio codec.

IX. COMPARISON WITH SIMILAR SYSTEMS

Recently, there have been two proposals for censorship circumvention that, similar to FreeWave, use the openness of VoIP to evade censorship. Due to their similarity with FreeWave we describe the advantages of FreeWave over them in this section.

A. SkypeMorph

SkypeMorph [28] is a pluggable transport [24] for Tor. SkypeMorph is designed to obfuscate the connections between Tor [8] users and Tor bridges [15] so that they look like legitimate Skype traffic. The main goal of SkypeMorph is to make it hard for a censor to distinguish between obfuscated Tor bridge connections and actual Skype calls using deep-packet inspection and statistical traffic analysis. A big implementation-wise difference with our proposal is that SkypeMorph does not completely run, but mimics, Skype, whereas FreeWave runs the target VoIP protocol in its entirety. FreeWave has the following main advantages over SkypeMorph:

Server obfuscation: Similar to the most of existing obfuscation-based techniques, SkypeMorph only provides traffic obfuscation, but it does not provide server obfuscation. A censor may not be able to identify SkypeMorph traffic through statistical analysis, since SkypeMorph shapes it to look like a regular Skype traffic. However, if a censor discovers the IP address of a SkypeMorph Tor bridge, e.g., through bridge enumeration [16], [17], SkypeMorph's obfuscations does not provide any protection since the censor can easily block its traffic by IP addresses matching. As an indication to the severity of this problem, the Chinese censors were able to enumerate *all* bridges in under a month [30]. Once a Tor bridge is known to a censor, SkypeMorph is not able to provide *any* protection.

On the other hand, FreeWave provides server obfuscation in addition to traffic obfuscation. Instead of morphing the traffic into VoIP, FreeWave uses the overlay network of VoIP systems to route the connections among users and servers. As a result, FreeWave's VoIP traffic gets relayed by "oblivious" VoIP nodes, hiding the identity (e.g., the IP address) of the FreeWave server. Even a censor who knows the IP address of a FreeWave server will not be able to identify and/or block client connections to that server, since these connections do not go directly to the server. For instance, if Skype is used by FreeWave the FreeWave connections get relayed by *Skype supernodes*, which are oblivious Skype users residing "outside" the censoring ISP (please see Section IV-B for further discussion). Note that there is not a one-to-one correspondence between supernodes and FreeWave servers, i.e., various supernodes relay traffic to a particular FreeWave server for different connections. As another example, if Google Voice is used by FreeWave, all the FreeWave connections get relayed by Google servers, hiding FreeWave servers' IP addresses. Note that we assume that VoIP connections are also encrypted.

Comprehensive traffic obfuscation SkypeMorph shapes Tor traffic into Skype calls, but it does not run the actual Skype protocol (except for the Skype login process) [28]. This can enable sophisticated attacks that can discriminate SkypeMorph from Skype by finding protocol details that are not properly imitated by SkypeMorph. For instance, SkypeMorph fails to mimic Skype's TCP handshake [56], which is essential to every genuine Skype call. Also, Skype protocol may evolve over time and SkypeMorph would need to follow the evolution. FreeWave, on the other hand, runs the actual VoIP protocol in its entirety, providing a more comprehensive traffic obfuscation.

No need to pre-share secret information: SkypeMorph

Pattern	FreeWave over Skype	Skype-Speak	Skype-Silent
Average packet rate (pps)	49.91	50.31	49.57
Average packet size	148.64	146.50	103.97
Minimum packet size	64	64	64
Maximum packet size	175	171	133

Table II
COMPARING COMMUNICATION PATTERNS OF REGULAR SKYPE WITH FREEWAVE-OVER-SKYPE.

needs to secretly share its Skype ID with its clients, as well as its IP address and port number (this can be done using Tor's BridgeDB [57] as suggested by the authors). Once this secret information is disclosed to a censor (e.g., through bridge enumeration) the identified Tor bridge will need to change both its IP address and its Skype ID, as suggested in [28], to reclaim its accessibility by clients. FreeWave, however, does not need to share *any* information with its clients: even the VoIP IDs of the FreeWave servers are publicly advertised without compromising the provided unobservability.

Obfuscation diversity: SkypeMorph is designed to morph traffic only into Skype. As a result, if a censor decides to block Skype entirely SkypeMorph will be blocked as well. FreeWave, on the other hand, is a general infrastructure and can be realized using a wide selection of VoIP services. Needless to say, SkypeMorph may also be modified to mimic other popular VoIP services, but it requires substantial effort in understanding and analyzing the candidate VoIP system. FreeWave, however, can be used with *any* VoIP service without the need for substantial modifications.

B. CensorSpoofer

A key goal in the design of CensorSpoofer [31] is to provide unobservability, as is the case in FreeWave. CensorSpoofer decouples upstream and downstream flows of a connection; the upstream flow, which is supposed to be low-volume, is steganographically hidden inside instant messages (IM) or email messages that are sent towards the secret IM or email addresses of the CensorSpoofer server. The IM IDs or the email addresses of the CensorSpoofer server need to be shared securely with clients through out-of-band channels. The CensorSpoofer server sends the downstream flow of a connection by spoofing a randomly chosen IP address, in order to obfuscate its own IP address. This spoofed flow is morphed into an encrypted VoIP protocol to obfuscate traffic patterns as well. A CensorSpoofer client also needs to generate "dummy" packets towards the spoofed IP address to make the connection look bidirectional. FreeWave makes the following contributions over CensorSpoofer:

No invitation-based bootstrapping: A new CensorSpoofer client needs to know a *trusted* CensorSpoofer client in order to bootstrap [31]. The trusted client helps the new client to send her personalized upstream ID and SIP ID to the CensorSpoofer server. Finding an existing, trusted CensorSpoofer client might be challenging for many new clients unless CensorSpoofer is widely deployed. Also note that even an existing CensorSpoofer client needs to re-bootstrap its CensorSpoofer connectivity if her personalized CensorSpoofer IDs are discovered by the censors. FreeWave, on the other hand, does not require an invitation-based bootstrapping.

Comprehensive traffic obfuscation Unlike FreeWave and similar to SkypeMorph, CensorSpoofer does not entirely run the VoIP protocol. This can enable sophisticated attacks that are able to find protocol discrepancies between CensorSpoofer and genuine VoIP traffic. Also, the use of IP spoofing by CensorSpoofer may enable active traffic analysis attacks that manipulate its downstream VoIP connection and watch the server's reaction.

Bidirectional circumvention: In CensorSpoofer VoIP connections only carry the downstream part of a circumvented connection. The upstream data are sent through *low-capacity* steganographic channels inside email or instant messages [31]. FreeWave, however, provides a high-capacity channel for both directions of a circumvented connection.

X. OTHER RELATED WORK

Censorship circumvention systems have been evolving continuously to keep up with the advances in censorship technologies. Early circumventions systems simply used network proxies [58] residing outside censorship territories, trying to evade the simple IP address blocking and DNS hijacking techniques enforced by pioneer censorship systems. Examples of such proxy-based circumvention tools are DynaWeb [6], Anonymizer [9], and Freenet [59].

Proxy-based circumvention tools lost their effectiveness with the advent of more sophisticated censorship technologies such as deep-packet inspection [2], [3]. Deep-packet inspection analyzes packet contents and statistics looking for deviations from the censor's regulations. This has led the circumvention tools to correspondingly sophisticate their techniques to remain accessible to their users. Many circumvention designs seek availability by sharing some *secret* information with their users so that their utilization is unobservable to the censors agnostic to this secret information. In Infranet [5], for instance, a user needs to make a special, secret sequence of HTTP requests to an Infranet server to

request for censored web contents, which are then sent to him using image steganography. Collage [7] similarly bases its unobservability on sharing secrets with its clients. A Collage client and the Collage server secretly agree on some user-generated content sharing websites, e.g., flickr.com, and use image steganography to communicate through these websites. The main challenge for these systems, which rely on pre-sharing secret information, is to be able to share secret information with a large set of actual users while keeping them secrets from censors; this is a big challenge to solve as indicated in several researches [18]–[20]. Sharing secret information with users has also been adopted by the popular Tor [8] anonymity network. The secret information here are the IP addresses of volunteer Tor relays, known as Tor bridges [15], that proxy the connections of Tor clients to the Tor network. This suffers from the same limitation as censors can pretend to be real Tor users and gradually identify a large fraction of Tor bridges [16], [17], [29].

More recently, several researches propose to build circumvention into the Internet infrastructure [10], [11], [51]. Being built into the Internet infrastructure makes such circumvention highly unobservable: a client's covert communication with a censored destination appears to the censor to be a benign connection to a non-prohibited destination. Telex [10], Cirripede [11] and Decoy Routing [51] are example designs using such infrastructure-embedded approach. Decoy Routing needs to share secrets with its clients using out-of-band channels, whereas Telex and Cirripede share the secret information needed to initialize their connections using covert channels inside Internet traffic. Cirripede uses an additional client registration stage performed steganographically, distinguishing it from the other designs. Even though these systems are a large step forward in providing unobservable censorship circumvention their practical deployment is not trivial as they need to be deployed by a number of real-world ISPs that will make software/hardware modifications to their network infrastructures, posing a substantial deployment challenge.

Another research trend uses traffic obfuscation to make circumvented traffic unobservable. Appelbaum et al. propose a platform that allows one to build protocol-level obfuscation plugins for Tor, called *pluggable transports* [24]. These plugins obfuscate a Tor client's traffic to Tor bridges by trying to remove any statistical/content pattern that identifies Tor's traffic. Obfsproxy [25], the pioneer pluggable transport, removes all content identifiers by passing a Tor client's traffic through an additional layer of stream cipher encryption. Obfsproxy, however, does not disguise the statistical patterns of Tor's traffic. SkypeMorph [28] and StegoTorus [27] attempt to remove Tor's statistical patterns as well by morphing it into popular, uncensored Internet protocols such as Skype and HTTP. Flashproxy [60] is another recently designed pluggable transport that chops a Tor client's traffic into multiple connections, which are proxied by web browsers rendering volunteer websites.

CensorSpoofer [31] is another recent proposal that, similar to SkypeMorph [28], shapes Tor traffic into VoIP protocols. CensorSpoofer is unique in separating the upstream and downstream flows of a circumvented connection, and in using IP spoofing to obfuscate its server's identity. A security concern with morphing approaches [27], [28], [31], [61] is that they do not provide a provable indistinguishability; censors may be able to devise advanced statistical classifiers and/or protocol identifiers to find discrepancies between a morphed traffic and genuine connections. Another approach that similarly uses VoIP traffic is TranSteg [62]; it re-encodes a VoIP call packets using a different, lower-rate codec in order to free a portion of VoIP packet payloads, which are then used to send a low-bandwidth hidden traffic.

XI. Limitations and Recommendations

Server location: In order to achieve server obfuscation special care needs to be taken in setting up a FreeWave server. In the case of Skype, for instance, the FreeWave server should be completely firewalled such that its Skype traffic is completely handled by Skype supernodes. Also, a FreeWave server should use a large, dynamic set of supernodes (i.e., by flushing its supernode cache [34], [35]) so that one cannot map a FreeWave server to its supernodes. A corrupt supernode (e.g., controlled by the censors) used by a FreeWave server can identify the clients that used FreeWave through that supernode. The mechanisms to protect server obfuscation vary depending on the utilized VoIP system.

Traffic analysis: If the VoIP service deployed by FreeWave uses a variable-length audio codec, like SILK [55], FreeWave's traffic might be subject to traffic analysis. In Section VIII-C, we showed that the current deployment of FreeWave over Skype performs well against simple traffic analysis, yet more sophisticated traffic analysis [63] may be able to distinguish FreeWave's current prototype from Skype. A trivial countermeasure is to add some pre-recorded human speech to FreeWave's audio, which would further reduce FreeWave's data rate. A better approach is to encode FreeWave's traffic into video, instead of audio, which is more robust to traffic analysis and provides much higher throughputs.

Trusting the VoIP provider: A VoIP provider colluding with censors can significantly degrade FreeWave's obfuscation promises if FreeWave deploys it. On the bright side, FreeWave can choose from a wide range of VoIP providers. In the case of Skype, in particular, Chinese Skype users get provided with a special implementation of Skype, TOM-Skype, which is suspected [64] to have built-in surveillance functionalities such as text message filtering [65]–[68].

Denial of service: Since FreeWave's VoIP IDs are public censors can exhaust FreeWave servers by making many FreeWave connections. Different approaches can be taken to limit the effect of such attempts such as the existing

sybil defense mechanisms [69], as well as usage limitation enforcement per VoIP caller.

XII. Conclusions

In this paper, we presented FreeWave, a censorship circumvention system that is highly unblockable by censors. FreeWave works by modulating a client's Internet traffic inside the acoustic signals that are carried over VoIP connections. Being modulated into acoustic signals, as well as the use of encryption makes FreeWave's VoIP connections unobservable by a censor. By building a prototype implementation of FreeWave we show that FreeWave can be used to achieve connection bit rates that are suitable for normal web browsing.

Acknowledgment

The authors would like to thank Prof. Vitaly Shmatikov, Qiyan Wang, the Hatswitch group at UIUC, and anonymous reviewers for their insightful comments on different versions of this paper. Amir Houmansadr and Nikita Borisov were supported in part by the National Science Foundation grant CNS 0831488 and the Boeing Trusted Software Center at the Information Trust Institute at the University of Illinois. Thomas Riedl and Andrew Singer were supported in part by the department of the Navy, Office of Naval Research, under grant ONR MURI N00014-07-1-0738 and by the National Science Foundation under grant number ECCS-1101338.

References

[1] "Arab spring: an interactive timeline of Middle East protests," http://www.guardian.co.uk/world/interactive/2011/mar/22/middle-east-protest-interactive-timeline.

[2] "Defeat Internet Censorship: Overview of Advanced Technologies and Products," http://www.internetfreedom.org/archive/Defeat_Internet_Censorship_White_Paper.pdf, Nov. 2007.

[3] C. S. Leberknight, M. Chiang, H. V. Poor, and F. Wong, "A taxonomy of Internet censorship and anti-censorship," http://www.princeton.edu/~chiangm/anticensorship.pdf, 2010.

[4] "Iran: Blogger died as a result of 'shock'," http://times247.com/articles/iran-blogger-died-as-a-result-of-shock, Nov. 2012.

[5] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, "Infranet: Circumventing Web Censorship and Surveillance," in *11th USENIX Security Symposium*, 2002, pp. 247–262.

[6] "DynaWeb," http://www.dongtaiwang.com/home_en.php.

[7] S. Burnett, N. Feamster, and S. Vempala, "Chipping Away at Censorship Firewalls with User-Generated Content," in *USENIX Security Symposium*, 2010, pp. 463–468.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *USENIX Security Symposium*, 2004.

[9] J. Boyan, "The Anonymizer: Protecting User Privacy on the Web," *Computer-Mediated Communication Magazine*, vol. 4, no. 9, Sep. 1997.

[10] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman, "Telex: Anticensorship in the Network Infrastructure," in *20th Usenix Security Symposium*, 2011.

[11] A. Houmansadr, G. T. K. Nguyen, M. Caesar, and N. Borisov, "Cirripede: Circumvention Infrastructure using Router Redirection with Plausible Deniability," in *ACM Conference on Computer and Communications Security*, 2011.

[12] P. Winter and S. Lindskog, "How China Is Blocking Tor," http://arxiv.org/abs/1204.0447, Tech. Rep. Arxiv preprint, arXiv:1204.0447, 2012.

[13] "Italy Censors Proxy That Bypasses BTjunkie and Pirate Bay Block." http://torrentfreak.com/italy-censors-proxy-that-bypasses-btjunkie-and-pirate-bay-block-110716.

[14] "Tor partially blocked in China," https://blog.torproject.org/blog/tor-partially-blocked-china, Sep. 2007.

[15] R. Dingledine and N. Mathewson, "Design of a blocking-resistant anonymity system," The Tor Project, Tech. Rep., Nov. 2006.

[16] D. McCoy, J. A. Morales, and K. Levchenko, "Proximax: A Measurement Based System for Proxies Dissemination," in *Financial Cryptography and Data Security*, 2011.

[17] J. McLachlan and N. Hopper, "On the risks of serving whenever you surf: vulnerabilities in Tor's blocking resistance design," in *the 8th ACM workshop on Privacy in the electronic society*, 2009.

[18] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger, "Thwarting Web Censorship with Untrusted Messenger Discovery," in *the 3rd International Workshop on Privacy Enhancing Technologies*, 2003.

[19] M. Mahdian, "Fighting Censorship with Algorithms," in *Fun with Algorithms*, ser. Lecture Notes in Computer Science, P. Boldi and L. Gargano, Eds. Springer, 2010, vol. 6099, pp. 296–306.

[20] Y. Sovran, A. Libonati, and J. Li, "Pass it on: Social networks stymie censors," in *the 7th International Conference on Peer-to-peer Systems*, Feb. 2008.

[21] "Ultrasurf," http://www.ultrareach.com.

[22] "Psiphon," http://psiphon.ca/.

[23] J. Appelbaum, "Technical analysis of the Ultrasurf proxying software," https://media.torproject.org/misc/2012-04-16-ultrasurf-analysis.pdf, The Tor Project, Tech. Rep., 2012.

[24] J. Appelbaum and N. Mathewson, "Pluggable transports for circumvention," https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/180-pluggable-transport.txt.

[25] N. Mathewson, "A simple obfuscating proxy," https://www.torproject.org/projects/obfsproxy.html.en.

[26] "Pluggable Transports Roadmap," https://www.cl.cam.ac.uk/~sjm217/papers/tor12pluggableroadmap.pdf.

[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 *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[28] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "SkypeMorph : Protocol Obfuscation for Tor Bridges," in *ACM Conference on Computer and Communications Security*, 2012.

[29] T. Wilde, "Knock Knock Knockin on Bridges Doors," https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors, 2012.

[30] "Ten ways to discover Tor bridges," https://blog.torproject.org/blog/research-problems-ten-ways-discover-tor-bridges.

[31] Q. Wang, X. Gong, G. T. K. Nguyen, A. Houmansadr, and N. Borisov, "CensorSpoofer: Asymmetric Communication with IP Spoofing for Censorship-ResistantWeb Browsing," in *ACM Conference on Computer and Communications Security*, 2012.

[32] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper, "Routing Around Decoys," in *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[33] S. Baset and H. Schulzrinne, "Skype relay calls: Measurements and Experiments," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[34] S. A. Baset and H. G. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," in *the 25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–11.

[35] S. Guha, N. Daswani, and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *the 5th International workshop on Peer-To-Peer Systems*, 2006.

[36] "VoIP growing statistics," http://www.sipnology.com/company/20-voip-growing-statistics.

[37] "VoIP Penetration Forecast to Reach 79% of US Businesses by 2013," http://www.fiercewireless.com/press-releases/voip-penetration-forecast-reach-79-us-businesses-2013, 2010.

[38] "The Importance of VoIP," http://ezinearticles.com/?The-Importance-of-VoIP&id=4278231.

[39] R. Koetter, A. C. Singer, and M. Tchler, "Turbo Equalization," *IEEE Signal Processing Mag*, vol. 21, pp. 67–80, 2004.

[40] A. Banerjee, D. J. Costello, Jr., T. E. Fuja, and P. C. Massey, "Bit Interleaved Coded Modulation Using Multiple Turbo Codes," in *IEEE International Symposium on Information Theory*, 2002, p. 443.

[41] C. Jones, "The Importance Of VoIP And A Business Continuity Plan For Business Survival," http://www.tech2date.com/the-importance-of-voip-and-a-business-continuity-plan-for-business-survival.html, 2011.

[42] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, Jun. 1999.

[43] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "RFC 1928: SOCKS Protocol Version 5," Apr. 1996.

[44] "Skype grows FY revenues 20%, reaches 663 mln users," http://www.telecompaper.com/news/skype-grows-fy-revenues-20-reaches-663-mln-users, 2011.

[45] H. Xie and Y. R. Yang, "A Measurement-based Study of the Skype Peer-to-Peer VoIP Performance," in *6th International workshop on Peer-To-Peer Systems*, 2007.

[46] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johanston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "RFC 3261: SIP Session Initiation Protocol," Jun. 2002.

[47] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet RFC 3550, Jul. 2003.

[48] M. Baugher, E. Carrara, D. A. McGrew, M. Naslund, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," Internet RFC 3711, Mar. 2004.

[49] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, and M. Bhatia, "Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments," Internet RFC 5853, Apr. 2010.

[50] "How to Configure SIP and NAT," http://www.linuxjournal.com/article/9399.

[51] 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 *1st USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[52] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Groups," in *the 3rd ACM Conference on Computer and Communications Security*. ACM, Mar. 1996, pp. 31–37.

[53] "VoIP codecs," http://www.en.voipforo.com/codec/codecs.php.

[54] "Google talk call signaling," https://developers.google.com/talk/call_signaling#Supported_Media_Types.

[55] S. Jensen, K. Vos, and K. Soerensen, "SILK speech codec," Working Draft, IETF Secretariat, Fremont, CA, USA, Tech. Rep. draft-vos-silk-02.txt, Sep. 2010.

[56] D. Adami, C. Callegari, and S. Giordano, "A Real-Time Algorithm for Skype Traffic Detection and Classification," in *the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking*, 2009.

[57] "Tor BridgeDB," https://gitweb.torproject.org/bridgedb.git/tree.

[58] I. Cooper and J. Dilley, "Known HTTP Proxy/Caching Problems," Internet RFC 3143, Jun. 2001.

[59] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley, "Protecting Free Expression Online with Freenet," *IEEE Internet Computing*, vol. 6, no. 1, pp. 40–49, 2002.

[60] D. Fifield, N. Hardison, J. Ellithrope, E. Stark, R. Dingledine, D. Boneh, and P. Porras, "Evading Censorship with Browser-Based Proxies," in *Privacy Enhancing Technologies Symposium (PETS)*, 2012.

[61] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis," in *Network and Distributed System Security Symposium*. The Internet Society, Feb. 2009.

[62] W. Mazurczyk, P. Szaga, and K. Szczypiorski, "Using Transcoding for Hidden Communication in IP Telephony," Arxiv preprint, arXiv:1111.1250, Tech. Rep., 2011.

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

[64] J. Knockel, J. R. Crandall, and J. Saia, "Three Researchers, Five Conjectures: An Empirical Analysis of TOM-Skype Censorship and Surveillance," in *the 1st USENIX Workshop on Free and Open Communications on the Internet*, 2011.

[65] "Dynamic Internet Technology Inc. Alleges Skype Redirects Users in China to Censorware Version-Ten Days After Users Are Able To Download Freegate Software Through Skype," http://www.businesswire.com/news/home/20070924006377/en/Dynamic-Internet-Technology-Alleges-Skype-Redirects-Users, 2007.

[66] "Surveillance of Skype messages found in China," International Herald Tribune, Tech. Rep., 2008.

[67] T. Claburn, "Skype Defends VoIP IM Monitoring In China," http://www.informationweek.com/news/210605439, 2008.

[68] "Skype says texts are censored by China," http://www.ft.com/cms/s/2/875630d4-cef9-11da-925d-0000779e2340.html#axzz1tpnzbzkm, Mar. 2009.

[69] N. Borisov, "Computational puzzles as sybil defenses," in *IEEE International Conference on Peer-to-Peer Computing*, 2006, pp. 171–176.