Modifying Network Traffic Profiles to Counter Recognition

By

Salman Haider

00000171616

Supervisor

Dr. Syed Taha Ali

Department of Electrical Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In

School of Electrical Engineering and Computer Science,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(July, 2020)

Approval

It is certified that the contents and form of the thesis entitled "**Modifying Network Traffic Profiles to Counter Recognition**" submitted by **Salman Haider** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Syed Taha Ali**

Signature: [illegible]

Date: 11-Aug-2020

Committee Member 1: **Dr. Fahad Ahmed Khan**

Signature: [illegible]

Date: 11-Aug-2020

Committee Member 2: **Dr. Arsalan Ahmed**

Signature: [illegible]

Date: 11-Aug-2020

Committee Member 3: **Dr. M. Imran Abeel**

Signature: [illegible]

Date: 11-Aug-2020

Thesis Acceptance Certificate

Certified that final copy of MS thesis written by **Salman Haider** , Registration No. 00000171616 , of Department of Computing, SEECS has been vetted by undersigned, found complete in all respects as per NUST Statutes and Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Name of Advisor: **Dr. Syed Taha Ali**

Signature: [illegible]

Date: **11-Aug-2020**

Dedication

Dedicated to my beloved parents and my wife

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Salman Haider**

Signature: [illegible]

Acknowledgment

I am grateful to Allah Almighty for guidance and blessings. I would like to express my sincere gratitude to my advisor Dr. Dr. Syed Taha Ali and all committee members for the continuous support during my MS course and research work, for their patience, motivation, and immense knowledge.

This work would not have been possible without support of my parents. I would like to thank my parents and my wife.

Salman Haider

Table of Contents

List of Figures

Abstract

With the rapid advancement of networking techniques, network traffic analysis has become key part of network management. Network traffic detection creates open threats to privacy and confidentiality of sensitive data. Different encryption techniques are being used to hide network information. But encryption doesn't hide statistical properties of a packet, therefore, network traffic still can be classified by manipulating these properties. In our research, we proposed an approach in which we added additional data to WebRTC data channel associated with RTC peer connection that modified packet sizes. We modified network profile of live streams such as Audi Stream, Video Stream, and Screen Sharing Stream, that helped us to show false data of network traffic profile to fool detectors and censoring authorities to circumvent censorship. As WebRTC data channel is peer-to-peer (P2P) connection, so sharing data through it is also mimicking signalling server.

Chapter 1

Introduction

1.1 Introduction

With the increase of internet users, network services are increasing rapidly. As internet services are getting more common these days, security and privacy concerns has been raised. It has been observed, attackers find vulnerabilities in security, exploit networks, and sensitive data gets leaked. Recent reports show that, due to network security vulnerabilities, frequency of data leakage is continuously increasing.

Traffic analysis plays very important role in traffic management. Analyzing network traffic is a serious threat to people's online privacy and sensitive data. With the rise of technology, traffic analysis techniques getting more advanced day by day that can detect even encrypted traffic [1]. There are different software applications e.g. Tor [2], Virtual Private Network(VPN), SSH Tunnel etc. to defeat the censoring infrastructure of a network. As encryption doesn't change any packet's statistical properties of flow in a network like direction, arrival time, packet size [3], therefore, its really easy for different censoring techniques to analyze and get important and sensitive information from encrypted traffic. Another limitation with censorship is, when a censor gets IP address that connects to the network, will be able to find the addresses of all connected people to that network.https://www.overleaf.com/project/5f232a245a3bd90001db4ed6

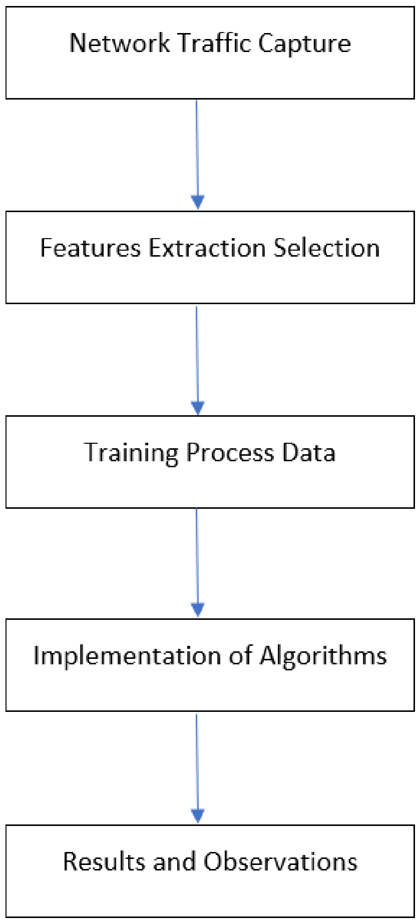


Figure 1.1: Network Traffic Classification Model

Statistical information of packet's flow help to extract traffic classification as shown in figure 1.1 and other important information related to encrypted traffic i.e. words or phrases can be extracted from Voice-over-IP using packet size or bit rate [4]. To avoid censorship and being detected, researches have designed systems e.g. SkypeMorph [5], StegoTorus [6], CensorSpoofer [7] etc. that help to hide information in their content by making proxy connection but these system has been failed due to inconsistent relationship between proxy protocol and cover protocol such as deocy routing [8–10].

There are different encrypted protocols that used to show structure and behavior of a packet in flow of network traffic. These information are extracted to classify the network flow that will help to detect traffic patterns. Before analysing network traffic, traffic flow should be identified either it is encrypted or not. For encrypted network traffic, Deep Packet Inspection (DPI) is most affected as compared to extracting statistical properties and behavior of packets. Traffic classification is used to find predefined categories of traffic i.e. normal or abnormal traffic. But this may contain also sensitive data related to network that maybe harmful in specific scenarios.

In this research, we will focus on modifying network traffic profile and shape. It will help to hide important and sensitive information about network traffic. By this methodology, we will secure and hide statistical information about packets. We applied it on multiple peer-to-peer (P2P) communications i.e. Text Chat, One-to-One Audio and Video Chat, Audio and Video Stream Broadcasting, and File sharing by modifying their data volume and packet size. It also helps to launch Denial-of-Service (DoS) attack and Distributed Denial of Service (DDoS) attack. As mostly censors cause leakage of traffic data, it will help to avoid censorship. Network censorship is practically applied by Governments [11] to control users access and block unethical activities.

1.2 Motivation

Network traffic is basically amount of data that moving through a network, encapsulated in the form of network packets. With the increase of users, network traffic has been rapidly increased and thus a lot of studies has been focused on network traffic analysis that is major part of network traffic management.

Whereas many techniques introduced to analyze network traffic and its properties to find anomalies and vulnerabilities. It increased privacy concerns of online data and users. With the time, different studies proposed encryption protocols and techniques to secure network traffic profile. After encryption, traffic is still being detected and classified using its different parameters. Because encryption doesn't affect different parameters e.g. arrival time, direction, number of packets etc.

FreeWave [12], SkypeMorh, StegoTorus connects with proxy network and Collage [13] encapsulates information while sharing content but due to censorship these systems has been failed due to different reasons e.g. inconsistency between decoy routing [14] and client or proxy. Client can establish connection to any host that is not blocked, and decoy routing then can help to connect any IP address (destination) that is blocked. Another reason for failure of these systems against censors is, cover protocol and proxy protocol behave differently to channel that cause inconsistency. Difference of characteristics of network traffic matched maybe the another reason of failure, as cover protocol and proxy protocol carries different characteristics [15]. Censorship may cause of leakage of sensitive information while analysing packet structure.

Network traffic profile is data volume or number of packets in a specific time duration that is called as Profiling Time Window (PTW).Network traffic measurement defines two categories normal traffic and abnormal traffic. Different approaches and protocols are used to detect this category that also create privacy concerns. Figure 1.2 is showing a traffic profile with sampling rate of 4 minutes. Network traffic profile also collects data during inactive periods, but that data is ignored extracting statistical properties. Extraction of data from traffic profile may cause of leakage of sensitive information about network that can lead to severe problems for organization.

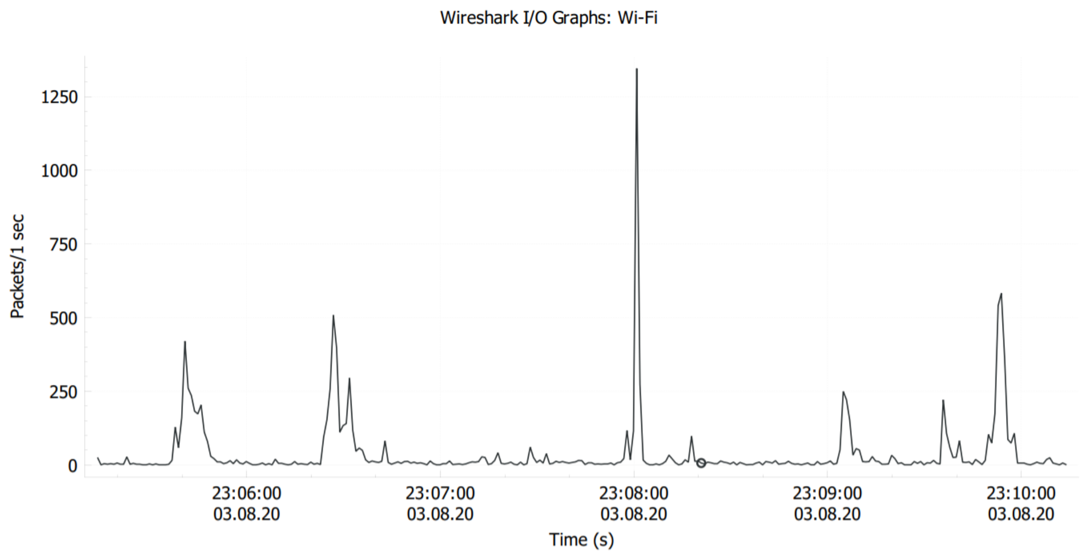


Figure 1.2: Sample Traffic Profile

1.3 Problem statement

Protection of privacy and sensitive data in real time network traffic is becoming the need of time. Networks and technologies are advancing day by day. Network traffic analysis causes a serious threat to the privacy of the network traffic and data crossing over it. To secure data, few researches have proposed to encrypt protocols and traffic data but still its possible to detect its pattern and formulate traffic class that cause revealing sensitive information about content and user's location using enumerated IP address of the connected users because encrypted traffic is having same statistical properties of packet. So, detectors can easily extract information to censor or classify network traffic.

The key challenge here is to avoid censorship and classification of network traffic. Basically, we need to modify the profile and shape of a network traffic that it can not be detected by any application. By reshaping traffic profile, it will make detectors to stop analysing correct data and they would not be able to classify traffic correctly. It will also help to lunching different kind of DoS or DDoS attacks.We need to find a model that should help in censorship circumvention and traffic detection by showing false data.

1.4 Research Objective

Main goal of this research is to find the most optimal solution of serious issues related to privacy and security of sensitive data about network traffic and its data. Network users want privacy about their physical and virtual location as well as their data transferring through network. Previous studies focused on encryption of network traffic and data to hide sensitive information related to network that can be helpful to exploit network traffic patterns. But using encryption techniques we can't hide all the statistical properties such as number of packets transferred over specific duration. This enables to detect traffic passing through network and they can manipulate it using different parameters. Extracting information about IP addresses of nodes on network can give information about their location that will breach security.

We will try to find its best solution that will help to fool detectors by giving them false data about network traffic. Our solution will show them invalid traffic provide to mislead them to classify the network traffic. There are different approached to classify network packets using different parameters but they do need of real and actual traffic profile to check categorically whether traffic is normal or not. It will also help to avoid censorship and to launch different attacks i.e. Dos, DDoS attacks etc. Our approach will also mimic socket server because we will modify network traffic that will be peer-to-peer (P2P) communication.

Chapter 2

Background Information

This chapter covers all the background information related to this research briefly. In 2.1 section their is a brief overview of Traffic Analysis. In section 2.2 WebRTC, free and open source framework for P2P communication, is discussed.

2.1 Traffic Analysis

Much studies has been focused on traffic analysis of a network using different parameters e.g. packet size, number of packets, virtual bit rate (VBR), arrival time etc. A lot of information can be extracted through traffic analysis. A traffic profile can be generated using this information. We can formulate that network traffic is normal or abnormal using traffic profile in a network.

Figure 2.1 shows that arrival rate of packets at a specific time. Using such information, we can make traffic prediction and classify that is this normal traffic or abnormal. Abnormal traffic normally can contain some malware or it can be a DoS attack. This information can also create privacy concerns for users. Detectors can extract much information about user including they

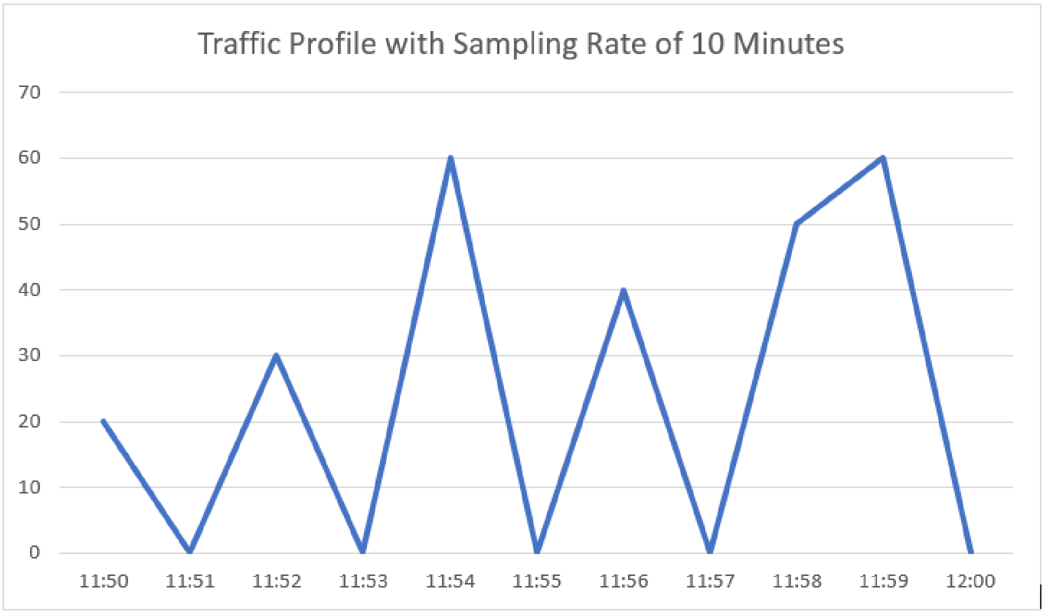


Figure 2.1: Network Traffic Profile

also can detect his location enumerating IP address of a packet header.

2.1.1 Conventional Packet Filtering

To extract traffic information through packet's header. Because firewalls had very low processing capacity so its difficult to process large data volume of packets. This technique is not enough for packet filtering, it is similar to getting a title of book without caring whats inside the book.

2.1.2 Deep Packet Inspection (DPI)

Deep Packet Inspection (DPI) is an advanced technique of network traffic filtering that used to detect content and signature of a packet by applying on application layer. Using DPI, network packets are examined by the ID of a packet (Signature). These packets are matched either with string or used different algorithms of expression matching [16] e.g. NIDS of Snort or L7-filter in Linux [17].

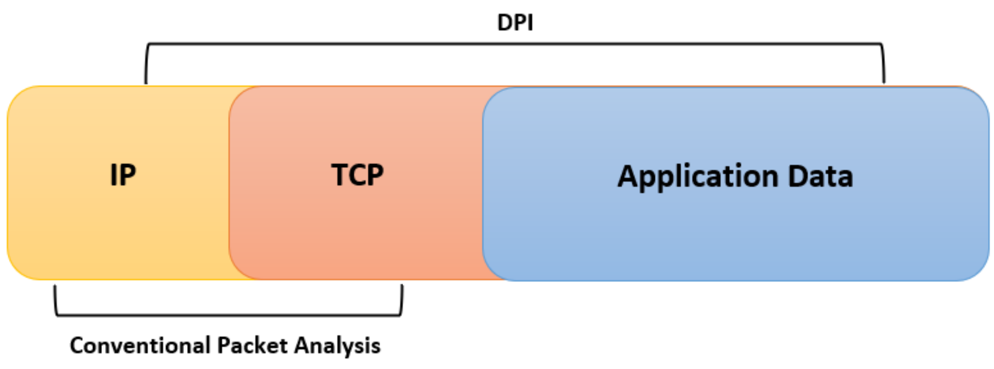


Figure 2.2: Deep Packet Inspection - An Insight

By inspecting traffic profile, DPI is also used to redirect specific packet to some different destination. DPI helps us to discover location and categorize traffic either is normal or not. We can also block or reroute network packets that different content or signature or those packets are not detected. DPI has few outcomes that makes it difficult to implement: as packets are collected in live stream so there is a large number of signatures i.e. Packet ID and defining its difficulty. As these days, different type of attacks has been launched so need to improve scalability of DPI.

2.1.3 Encrypted Traffic Classification

Traffic classification [18, 19] is the key of network model, management, and planning. There was privacy concerns of online data to analyze network traffic and its statistical properties. To overcome this limitation, different researches introduced different technologies and methods to encrypt network traffic that was a serious issue to network traffic classification. Peer-to-peer application are trying to defeat Internet service providers [20] by applying encryption and obfuscation techniques of protocols.

Internet users are more concerned these days about their privacy and security. Onion Routing [21] is also used for establishing connections anonymously to advance the privacy of online content shared. TLS, SKype, SSH, and BitTorrent, all these, protocols are using encryption techniques to provide privacy and confidentiality. Information security is threatened by dffer-ent encrypted channels as Trojans, Advance Persistent Threat (APT) [22] defeat firewalls to broadcast confidential data.

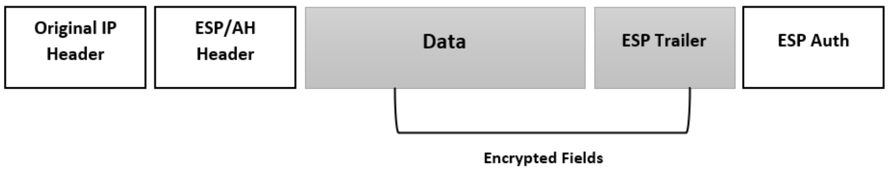


Figure 2.3: IPSec Packet Structure

Encrypted traffic doesn't provide security and privacy to online data. As we can see in image 2.3 that only data part is encrypted. Detectors can extract information about packets and its routing through packet header. Its destination location, size, and arrival time can also be retrieved that can be a challenge for encrypted protocols to preserve its security. So, encrypted traffic analysis can provide detection and forensics analysis of network traffic that can be misused.

2.2 WebRTC

WebRTC is a free and open source framework who provide real-time communication capabilities to web browsers. It supports audio, video and generic data to sent between peers. Media is only shared between peers without sharing to server. WebRTC is flexible and easy to use. It access media devices, opens own peer connection and starts discovering peers. After connecting to peers, it starts streaming media to all connected peers. By 2016 there was an estimate 2 Billion installed browsers works fine with WebRTC.

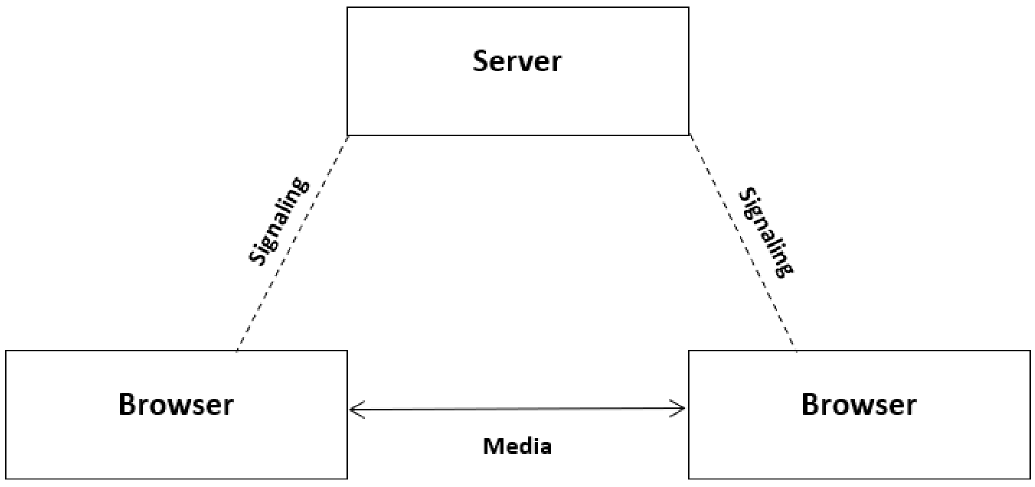


Figure 2.4: A simple WebRTC System

There are different web applications of WebRTC that utilizes different resources camera or microphone, and also applicable for advance way of communication such as video chat and screen sharing. Many platforms like Skype, Facebook, Google Hangouts use RTC [23] but they need plugins to have real-time communication. Downloading and installing a new plugin can be annoying for users. WebRTC supports all modern browsers e.g. Google Chrome, Opera, Firefox etc. without downloading and installing any plugin.

2.2.1 WebRTC API

WebRTC provides two standard technologies, capturing media devices and connection between peers (P2P). It facilitates both computers and smartphone. Both type of devices are supported with all kind of features such as capturing media, connecting peers, and streaming data. WebRTC is designed with three APIs (Application Programming Interface) implemented in JavaScript:

- MediaStream - MediaStream also known as getUserMedia captures input stream from media devices. The stream can be a video track, it may be generated by using hardware device or maybe pre-recorded video and screen sharing service, an audio track or other types such as files or text. Output of a media stream can be sent to more than one location. A stream allows the associated app to collect and manipulate data to specify outcome.

- RTCPeerConnection - RTCPeerConnection deals with the connection between different peers to communicate. Communication media can be different e.g. Audio, Video, Text Chat, File Sharing etc. In order to connect, peers need an ICE server configuration for signaling. ICE server can be TURN or STUN server. They basically provide ICE candidate to remote peer.

- RTCDataChannel - RTCDataChannel wroks with RTCPeerConnection API. It works with SCTP (Stream Control Transmission Protocol) [24]. RTCDataChannel can transmit text or a file or any encrypted data. It works with an interface associated with RTCPeerConnection which can be used to transfer bi-directional data between peers. Each peer can have up to 65,534 data channels.

Real time communication is very sensitive to time parameter, as audio and video stream is designed to tolerate packet loss. Application must be enough intelligent to handle packet loss or delayed recovery. As this is real time communication, therefore, UDP is preferred protocol for delivery of real time data. UDP supports real time communication in browsers but for WebRTC, browsers needs a lot of protocols and services above it shown in figure 2.5. UDP doesn't offer reliability and it delivers each packet to the destination app when it arrives.

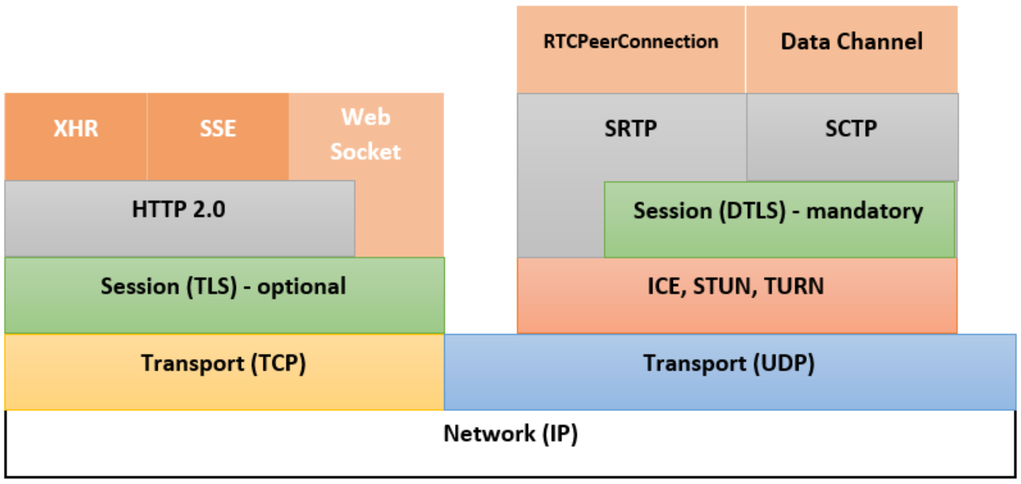


Figure 2.5: WebRTC Protocol Stack

2.2.2 Privacy and Security

MediaStream or getUserMedia is an important feature that only works in a secure context. Secure Context is a window which fulfill at least minimum standards of authentication and privacy [25]. Furthermore, before capturing media streams, user's permissions is also required for both audio and video inputs.

Chapter 3

Literature Review

Many studies have proposed that how data is can be extracted through a network traffic and how it can be used for different purposes such as censorship. Different researches also proposed different techniques of encryption to secure data but they still couldn't hide network traffic information as we can't hide packets statistical properties such as arrival rate, packet size and direction.

Mazurczyk [26] have presented a study that we can use Skype to transfer hidden information. They analyzed network traffic deeply and find out those packets in which there was no noise because there is high correlation between noise and packet size in Skype. They used encrypted data to transfer and shared secret of encryption technique on communication side through that it can be decrypted at destination. Its results were really promising that show that this staeganorgraphic approach can offer high bandwidth almost 1.8 kbps, utilizing 30% of silent packets. He also proposed that this approach SkyDe can also be used for other IP protocols to utilize their packets without noise. Another research [27] proposed similar concept of hiding data in Skype traffic. This research was based on the idea of intentional loss of packets to replace with payload of selected packets with encrypted data. They tested it on different variants and concluded that most optimal approach would be utlizing 1% of packets from the video packets for data hiding and transferring purposes. This approach results into higher bandwidth up to 0.93kbps and little quality degradation that can be ignored.

Abdullaziz [28] proposed a steganographic technique to hide data in UDP packets that is non-detectable. In this approach he proposed that the parity of 1's and 0's are utilized to exploit and encoded to secret data. This approach is based on UDP packet size that is totally random. He concluded that these channels might have low bandwidth but these channel and data shared on these channels cannot be detected. They also shared secret key between hosts using communication channel that is commenced between them. To covert the channel through which data is being shared, matched the data size of packets, if packet size is not matching is single byte padded. Such packets those are available are extended with additional source of secret data.

Another approach of analyzing un-observable network traffic [9] was proposed by Houmansadr. They analyze main social media that use VoIP data communication and try to find the result about hiding data and mimic the data. The findings are quite good, before doing any kind of data mimicking, must see the basic advisories protocol, before data hiding the someone had to find the flaw in the system which is really tough because system like Skype already have many ad-hoc security checks. The important finding was that wrong data hiding or data intrusion is much worse the no data hiding, IP can be tracked and blocked by doing that because system like SWEET and Skype have high level security that can block the IP after finding the irregularities in Network traffic. The last finding was quite discouraging for the data hiding as they preferred to not hide data in these types of system software. for the Mimicking side protocol is really tough as as they are really complex in correlations and dependencies. He proposed that he is not mimicking the protocol instead hiding data into protocol stack such as SWEET. [29] embeds secret data in email messages. Packets containing hidden data maybe statistically anomalous with comparison of normal Skype packets. On the other side, to detect such kind of anomalies traffic analyzing is required at large scale that increase the benchmark for censoring authorities. He also proposed that "unobservability by imitation" is totally a flawed approach so introduced his own communication approach. An approach of censorship circumvention [7] was also presented, they monitor the network with different types of techniques mainly are IP filtering and Packet inspection. Their system is highly suitable for detection unusual traffic on the network and can find the dummy IP with the help of system. They made infrastructure that covers the idea of decoupling the both channels e.g. upstream, downstream etc. Upstream channel can hide request data using some hiding technique like steganography within Email or Internet Messenger (IM) while downstream channel spoof IP address to conceal the real address of proxy server.

S Khattak [30] proposed an approach to support censorship resilient transport. Researches launched attacks to check censor's capabilities. The censorship model is in between client and server. They also proposed a model for a LC (Cirumvention System) that helps a censoring authority to communicate over network and introduced a system that can resist censorship by provide link circumventions. They main research on direct censorship and use the model of censor on client, sensor on server. to achieve the censorship in the model they have used Block routing information that connect the rows of the routing table, these rows mainly contain the source IP address and the Destination IP address. After detecting the IP address the Firewall block those IP addresses for 90 seconds. This approach can also fine the corrupt IP addresses from the routing table and its helps a lot in Hijacking and DNS manipulation. Li [15] proposed an approach to avoid censorship of video streaming named as Facet. Facet work in this flow; Facet client distributes ID for conferencing call for discovery. It can be public or private. Then Facet client sends connection request to server. After accepting this request by client, initial connection is established. After connecting with server, client sends URL of censored video to server. Server sends this URL to facet pipeline to download, decode and resize the video. After acceptance, client can watch the video in video conferencing session.

An approach was proposed [31], how to migitate DoS (Denial-of-Service) attack by analyzing network traffic. Researchers proposed the model that pass all request through Intrusion Detection System (IDS) to check whether the requests are normal or abnormal. If they find any anomaly, it seems, it may be a spam request to launch a DoS attack so all the information about that request is recorded as log in database. When the request comes to IDs, request data is checked into logs. If it exist there, then the request is blocked otherwise data is sent to server.

Herrero [32] proposed a communication model then modify datagrames in WebRTC, Proposed model communicate over a tunnel. There solution is consist of three main techniques and they are tunnel for data transmission, a local gateway for real time communication and lastly the Central gateway. It establishes a tunnel between client and server. System receives WebRTC data and converts it into frames, and sends them to tunnel. Through tunnel, enhancement features are added to transmit those frames to network. This technique is having four stages, in the first stage Web RTC create session protocol and it is encrypted with password, browser B receives that and sends the answer to the first server. In the second stage, as the communication channel is built, the server a sent IP address and the information related to ports and protocols. Browser B receives that IP address and protocol information. In the third step both the browsers complete their verification. In the fourth stage proper communication will occurs and UDP and TCP packet will be sent to each other. the information will be secured as the payload is encrypted. Fiat [33] proposed a peer-to-peer (P2P) Addressable Network for accessing n items in n number of nodes. Every search in networks take O(log n) time. His approach was to avoid censorship that even after removal of large number of nodes. they create content addressable memory that can be used in a completely distributed function. This is work very well in the loaded environment as it has the load balancing approach in the system, the reason is that they choose the bottom super node completely randomly in the system and so single bootom node will not be overloaded. This type of network is created in distributed environment. This system is highly spam resistant it means any node can complete control of all nodes in the network but still spam cannot be generated. Meidan [34] proposed an approach to analyze network traffic through IoT devices. He collected data from different devices and label for all different devices. He trained a classifier to detect that traffic is generated by IoT devices or not. After labelling stream, he associate each IoT devices to a class. After experiments, he proposed his approach has 99.2% accuracy. To test this model, he collected traffic data from PCs and Smartphones and this data was stored in pcap file. By extracting information from this file, packets were transformed to sessions with 4-tuple contains source, destination IP, and port numbers. Each session was presented by a vector from the transport and application layers features and enhanced with some publically available data set such as GeoIP. After

completing the data set with labels, data set was divided to three categories based on session classifiers, single-session, multi-session, test set to evaluate results.
Moghaddam [35] presented an idea, SkypeMorph, of prevention of detecting a specific type of network traffic such as Tor connections e.g. analysing traffic between a remote peer and peers in this network. Clients obfuscates the data to bridge. He used Skype Video calls as our target protocol to investigate. He proposed an approach to create serious issues for censoring authorities to distinguish between bridge connection and actual Skype calls. An advantage of SkypeMorph bridges easily change IP address and port number whitout sharing information to clients. SkypeKit let peers to share streaming data through the Skype network but this data is relayed by other nodes. It creates an extra overhead on the network, therefore, SkypeMorph sends data directly from client node to bridge.

Geddes [8] proposed that those systems that block detection to avoid censorship such as SkypeMorph, Censor Spoofer, and FreeWave use cover channels to hide proxy connections. He proposed that these system can still be attacked because of client proxy and loss intolerant. Results of this research proposed that, such protocols are not a good choice to avoid network traffic detection to avoid censorship. An approach [36] was presented that decoy routing doesn't need a peer to connect to specific IP address that is blocked on the network. It is possible for a peer to connect any unblocked host, decoy routing is used to connect to destination IP address that is blocked without sending any information to host. Different switches is used in different location in the world, they work as a decoy router for the system. The main framework is, the packet is sent by the user to the decoy switch and the packet had a message identify the hijacking. The main design have following steps, the decoy switch will have centralized management, suspected network traffic detection and mainly the diversion of traffic in a efficient way. Decoy routing need a friendly environment of network. Elahi [37] proposed a model to avoid censorship in network traffic. User try to access CRs and other relevant information through dissemination channel or out pf bound channel. Then user can connect over the channel in which data is shared, and CRs acts as a proxy. The connection is may or may not be hidden to prevent detection. The proxy then redirects the user's request locally or some covert destinations. One of the biggest disadvantage of censor is that they can block any infrastructure but can not shut down. However, sometimes damage by blocking, is not so worst. Sometimes a censor can deploy more resources than expected to compromise network strategies that can cause of an attack. For instance, a censor utilizes a large pool of IP addresses that can cause vulnerabilities; audio/video streaming should be packet loss or delay tolerance while mail services can tolerate high latency. Such attacks are least harmful, can damage on specific protocols such as VoIP that is used only for video and audio network traffic. Kopsell and Hilling [38] presented solution to avoid censorship in network traffic by blocking settings, their classification was based on censor's decision making process that is valuable for this scenario. Decision making process was supported with communication based on TCP/IP layer. Such communication can be a stream, depends upon either it is encrypted or not. Information extracted through such information can tell us either its video or audio stream and these stream are communication in the form of bytes that can be detected. Two limitations of blocking are: structure of service and information distributed about that service. They mentioned the example of connecting large set of access points and all are circumvention strategies through which censor is completely failed to detect.

Yu and Lee [39] proposed that currently all techniques who avoid censorship such as Tor totally depends upon volunteers who participate to run relays. The actual problem is to gather so many volunteers who work without any benefit. There are different techniques for volunteering such as e-Cash. They proposed a model to give benefits to both censored and uncensored users who utilize their own resources. They gave a model by using users browsing experience is optimized providing a practical relay in structure. Major goals of this research are, users should not be blocked by censor. Instead of this, they should be automatically disconnected from internet. User should not be identified by censors that they are using this approach or not. Another goal is that, it should not be detected that which users are using what content on network. This approach provides low latency like normal. It is implemented on common platforms and protocols such as TCP or TLS. This system contains multiple components such as browsing hosts, their goal is to target websites anonymously without being detected, flower hosts, to connect these websites anonymously, distributed servers that have capability for spoofing IP address. They evaluated from experiments that this approach provides a mechanism to avoid censorship and usability of user experience in daily web browsing. Different researches have proposed models to hide proxy connections with the use of decoy routing or cover protocols. They analyzed Skype-like communication to tunnel IP traffic through different proxies and fool different sessions [7, 12]. Frolov [40] developed a framework uTLS to mimic other TLS implementations. They used real time network traffic data and fool many TLS system without any extra manual effort. He collected 11B TLS client messages in time period of more that 9 months and analyzed them. They also collected and analyzed 5.9B messages of TLS servers. He tested existing approaches of to avoid censorship at risk of being blocked, and found that many of them are detectable. His developed library helped users to make fool many TLS handshakers that also allow to counter recognition and censorship. Researchers also made their data set to public to compare popular TLS fingerprints and their own devices TLS.

Chapter 4

Proposed Methodology

In previous chapters we have discussed, how network sensitive information can be leaked that creates privacy issues. Network traffic detection can also cause censorship that may block different proxy servers or even Tor. In previous researches, different encryption techniques has also been proposed to hide traffic data but encryption techniques cannot hide packet's statistical properties thus network traffic was still available that was a privacy concern. To overcome these limitations, we will try to find most optimal solution to modify network traffic profile that will help to avoid network traffic detection and censorship. We worked with peer-to-peer (P2P) communication using different sources such as Text Chat, Video Chat, Audio Chat, and File Sharing. We used modern, free, and open source framework WebRTC. As WebRTC supports all modern browsers, it was easy to configure and use. Another reason to use WebRTC is that it has a large number of audience. Many applications are using WebRTC.

To modify traffic profile, we added additional data through data channel to communicate. Data sending through data channel doesn't affect media stream or communication channel. When peers are discovered and connected

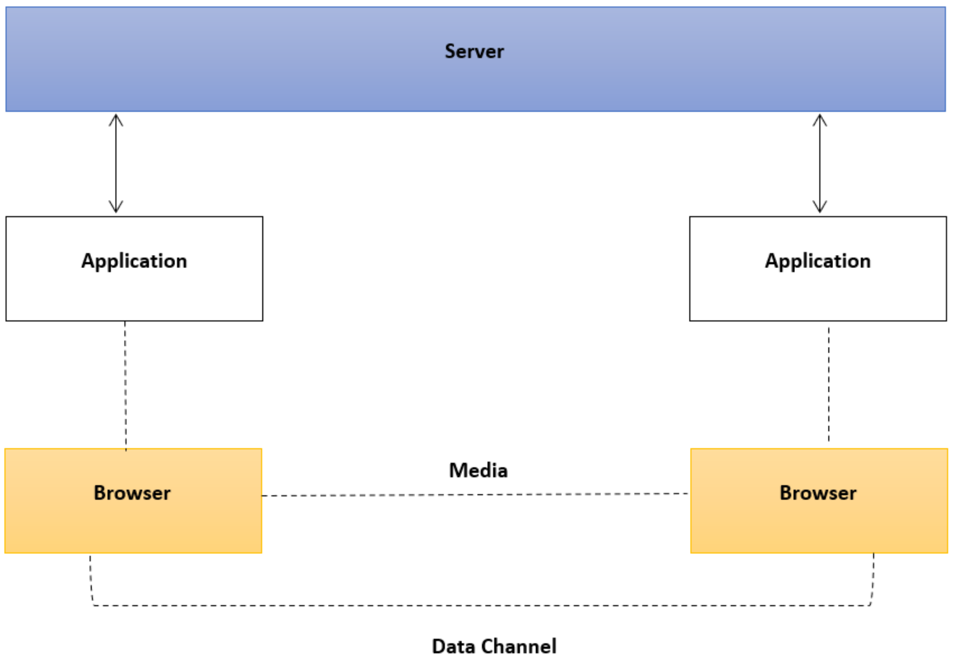


Figure 4.1: A simple WebRTC System

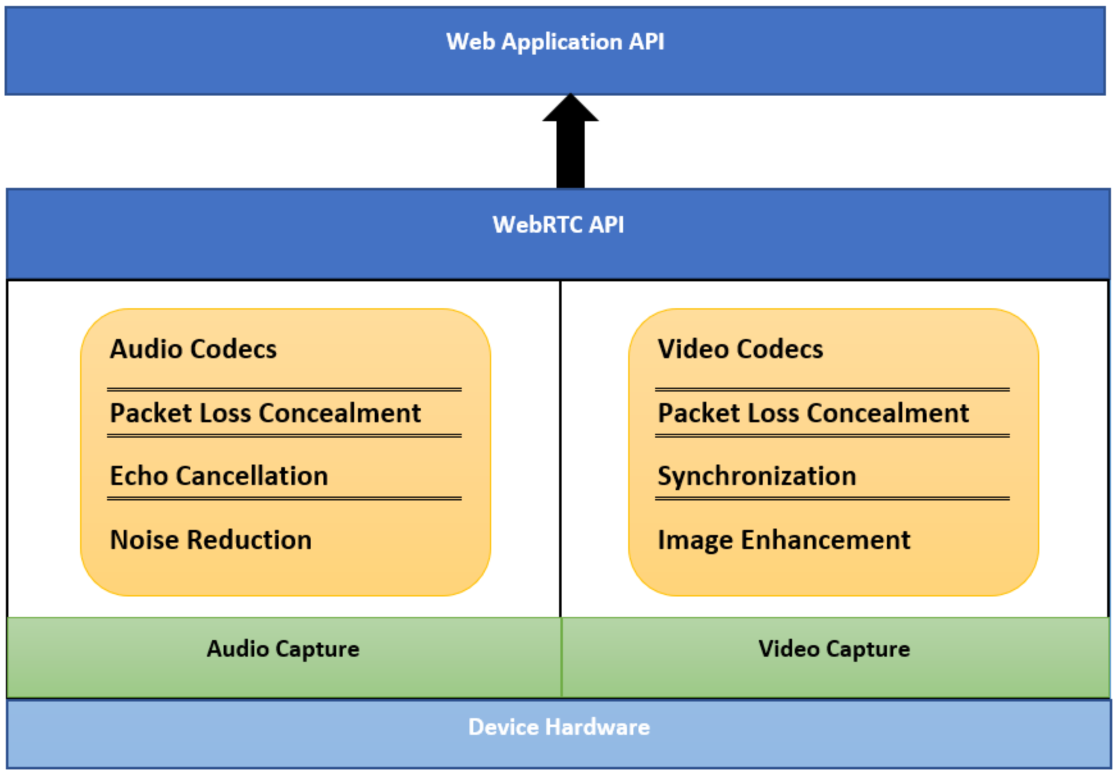


Figure 4.2: Capturing Media Audio/Video

data channel is created associated with that connection.

4.1 Media Stream

We are using different media stream to show network traffic such as Text Chat, Audio Chat, Video Chat, and File Sharing. Audio chat and Video chat depends upon hardware or if virtual devices are installed to stream pre-recorded audio or video track. Media streams in WebRTC is supported by all modern web browsers e.g. Firefox, Opera, Google Chrome etc.

Figure 4.2 is showing that how media is captured through hardware devices or virtually installed devices then WebRTC API manipulate data that is handled by web application.

4.2 RTC Peer Connection

RTCPeerConnection is an api that is fully responsible of all functions to discover and connect peers, and sharing streams between all peers. Streams can be of any type audio, video, file, text etc. It manages complete workflow of NAT (Network Adress Translation) traversal. NAT traversal is a networking technique of establishing and maintaining connections of all gateways that implement NAT. RTCPeerConnection helps to create connection offer, accept the offer, and retrieve the current state of that connection. It keeps track of local and remote streams. This API sends keepalives to STUN or TURN server between peers automatically.

4.3 RTC Data Channel

Data channel API allows to exchange arbitrary data between peers as Web Socket does but it send peer-to-peer (P2P). Each data channel is associated with a peer connection that provides reliable delivery of sent messages and out of order delivery sent messages. We can share different kind of data through data channel such as file or text etc. In our proposed work, we add some additional data through data channel. We tried with both text string and file. We also sent encrypted data through data channel and decrypted it on receiving peer. For encryption, we used AES algorithm.

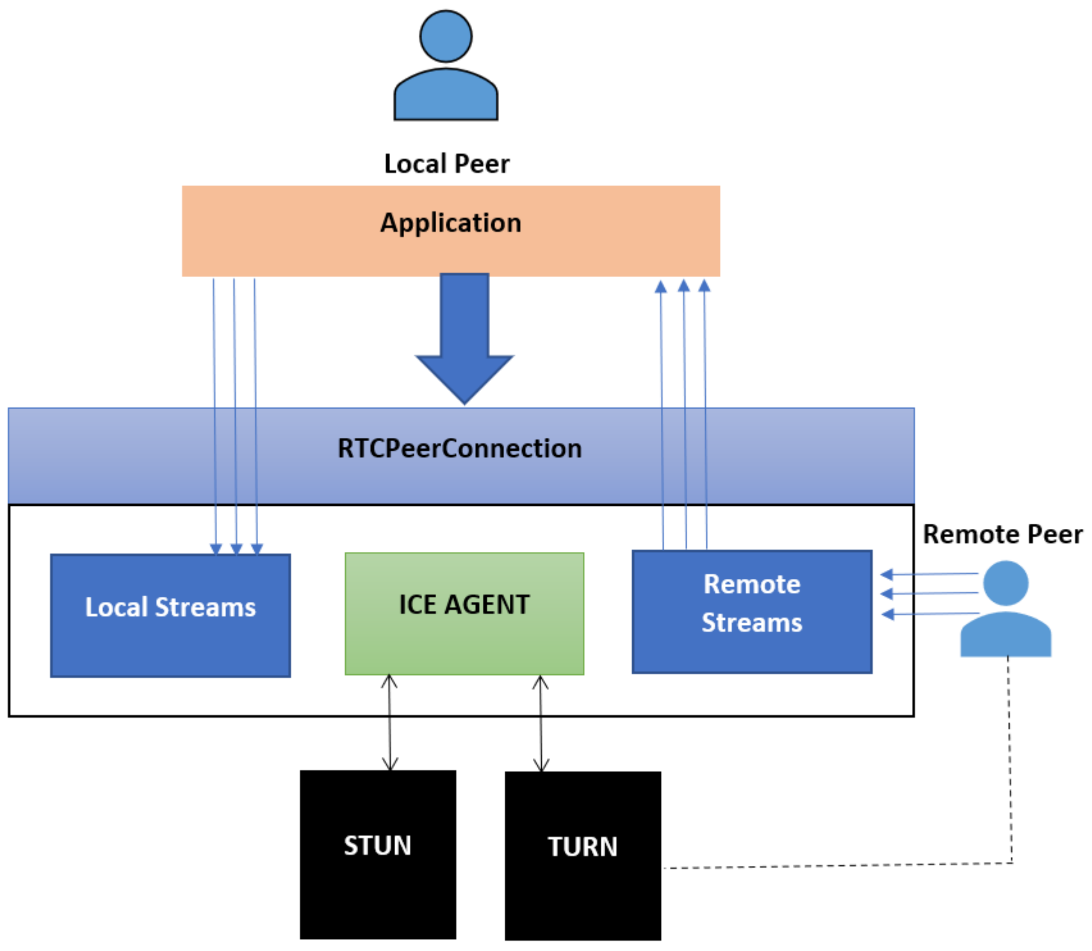


Figure 4.3: RTCPeerConnection API

Chapter 5

Implementation

As described in previous section, we implemented different modules of WebRTC to check traffic then we added additional data to modify traffic so that we can fool detectors to analyze network traffic by providing false data. For this purpose, we established a connection between two peers and created an associated channel to that peer and sent extra data so that we can compare network traffic captured earlier and later. In this section, we will see codes of creating connections, capturing media streams, sharing between peers, creating a data channel, and sending additional data through that channel to see results.

5.1 Creating Peer Connection

In order to implement our research, first of all we created a new connection so that we can discover another peer to connect and share media.

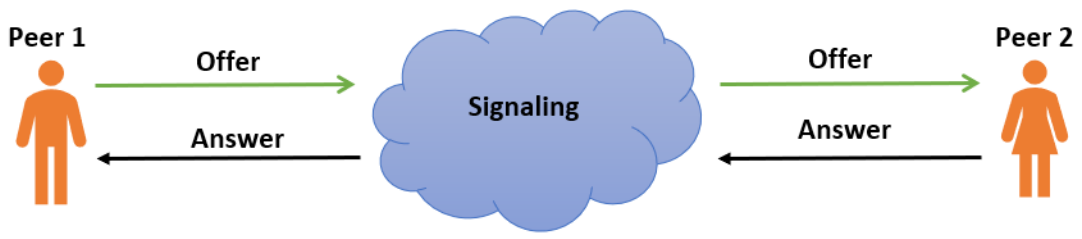


Figure 5.1: RTCPeerConnection API

```
// defining peers, we can also add servers as parameter, default is null
const localPeer = new RTCPeerConnection()
const remotePeer = new RTCPeerConnection()

//creating an offer using promises
localPeer.createOffer()
.then(offerSDP => {
    localPeer.setLocalDescription(offerSDP)
})

//creating an answer for offer
const remoteSDP = new RTCSessionDescription()
remotePeer.setRemoteDescription(remoteSDP)

remotePeer.createAnswer()
.then(answerSDP => {
    remotePeer.setLocalDescription(answerSDP)
})
```

5.2 Capturing Media Streams

We deployed our research with different media streams. In this section, we will see how we captured media streams. For audio and video streams, we used hardware devices e.g. microphone, web camera etc.

```
// for Audio Streams
const mediaConstraints = {
    audio: true,
    video: false
}

// for Video Streams
const mediaConstraints = {
    audio: false,
    video: true
}

// for Audio/Video stream
const mediaConstraints = {
    audio: true,
    video: true
}

// for Screen Sharing
const screen = {
    mandatory: {
        mediaSource: 'screen',
        maxWidth: 1366,
        maxHeight: 1080,
        minAspectRatio: 1.5
    },
    optional: []
}

const mediaConstraints = {
    audio: true, // audio can be muted
    video: screen
}



// getUserMedia API
const stream = navigator.mediaDevices.getUserMedia(mediaConstraints)

// adding stream to web interface
const media = document.createElement('video')
media.srcObject = stream
```

5.3 Creating Data Channel

DataChannel API is used to send arbitrary data between peers. In our research, we are using this API to send additional data to modify network.

We are using this API to mimic signalling or socket server. This data can not be traced, so it never can be calculated how much data is being sent through this channel. As we have already discussed that, data channel is associated with RTC peer connection, so we used already established peer connection in above code.

```
const options = {
    ordered: false,
    maxPacketLifeTime: 2000,
    maxRetransmits: 3
}
const dataChannel = localPeer.createDataChannel("channelName", options)
```

5.3.1 Adding Text Data

As we have created data channel, now we are going to add some text data. We used encryption algorithm AES-256 to encrypt text string using npm (node package module) package 'cryptr'.

```
// to encrypt
const encryptedText = cryptr.encrypt('Text String')
// e7b75a472b65bc4a42e7b3f78833a4d00040beba796062bf

// to open a data channel
dataChannel.onopen = () => {
    console.log('Channel is opened now')
}
```

```
// to send encrypted text
dataChannel.send(encryptedText)

// to receive data through data channel
dataChannel.onmessage = event => {
    console.log('Data', event.data)
}

// to decrypt
const text = cryptr.decrypt('e7b75a472b65b3a4d00040beba796062bf')

// to close data channel
dataChannel.onclose = () => {
    console.log('Channel is closed now')
}
```

5.3.2 Adding File Data

After adding text, we tested it with bigger amount of data. For this purpose, we added file. To add file in RTC Data Channel, we read it through FileReader API. We divided file into chunks.

```
// file input from HTML form
const file = files[0] //an array of multiple files

// File Reader API to read file data
const fileReader =  new FileReader()
fileReader.readAsDataURL(file)
fileReader.onload = (event, fileData) => {
    var data = {}
    const chunkLength = 2000
    if(event){
        fileData = event.target.result
    }
    if(fileData.length > chunkLength){
        data.stream = fileData.slice(0, chunkLength)
    }else{
        data.stream = fileData
        data.last = true
    }

    // to send file data
    dataChannel.send(data)
}
```

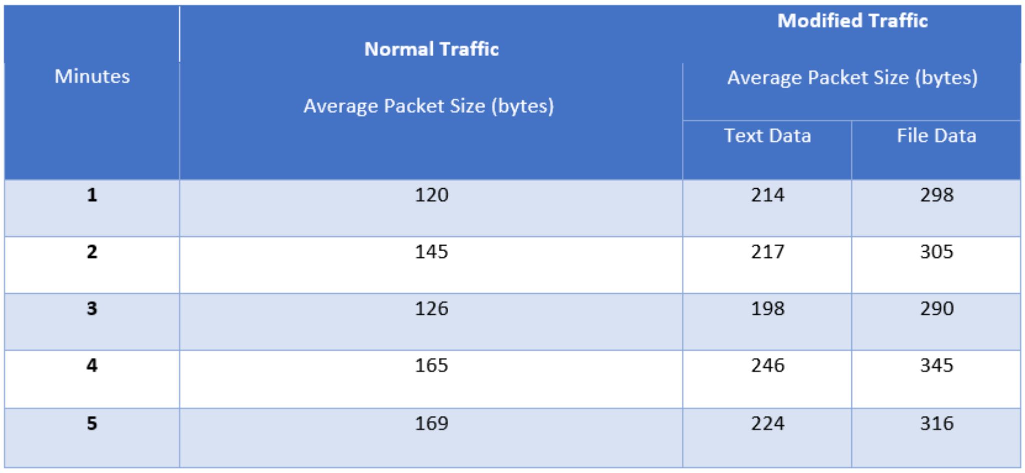
Chapter 6

Results Evaluation

We deployed three media sources used by WebRTC and appended data in it. We captured network traffic using Wireshark, taking 5 minutes as sample rate for all.

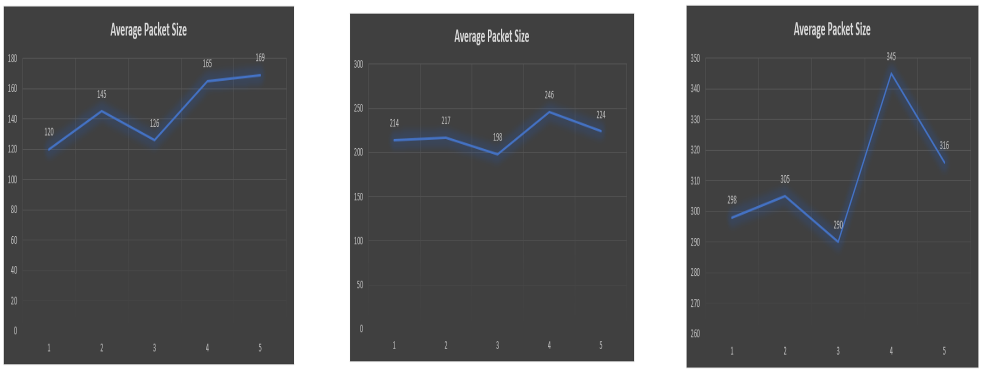
6.1 Audio Stream Results

We started an audio communication and connected with other peer. Initially we captured packet sizes for five minutes. After that we added, encrypted text data and file stream then checked packet sizes, details given below in figure 6.1.



Minutes	Normal Traffic	Modified Traffic	
	Average Packet Size (bytes)	Average Packet Size (bytes)	
		Text Data	File Data
1	120	214	298
2	145	217	305
3	126	198	290
4	165	246	345
5	169	224	316

Figure 6.1: Average Packet Size of all Audio Streams



(a) Normal Traffic (b) With Text Data (c) With File Stream

Figure 6.2: Network Traffic for Audio Streams

Figure 6.2 shows that how much network traffic is modified in audio streams with adding some additional data. First we captured packets for five minutes, average packet size was 145 bytes, after adding encrypted text stream for 5 minutes, packet size rose to 220 bytes and by adding more data such as file stream, it reached to 311 bytes on average. We ignored silent packets or having no data in it.

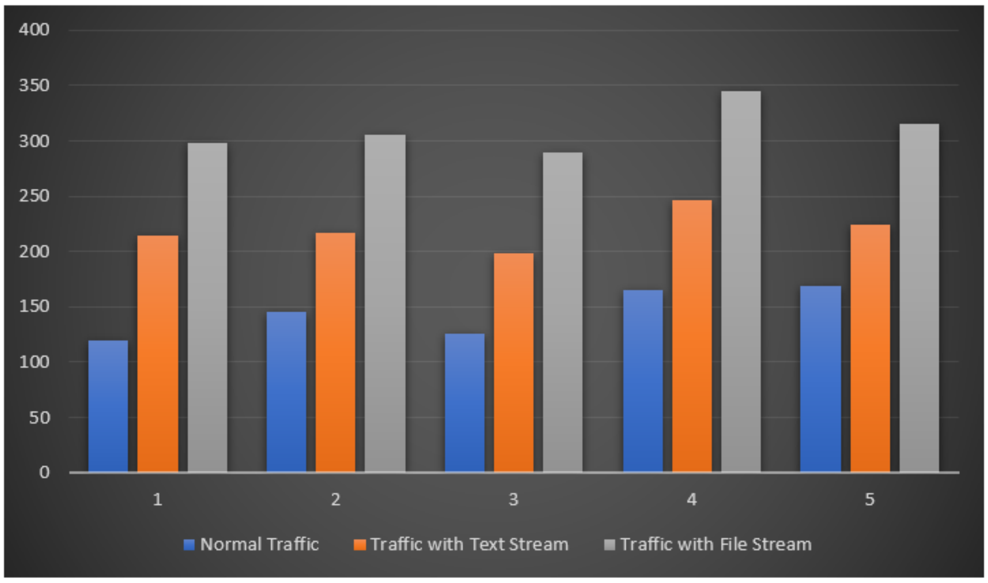
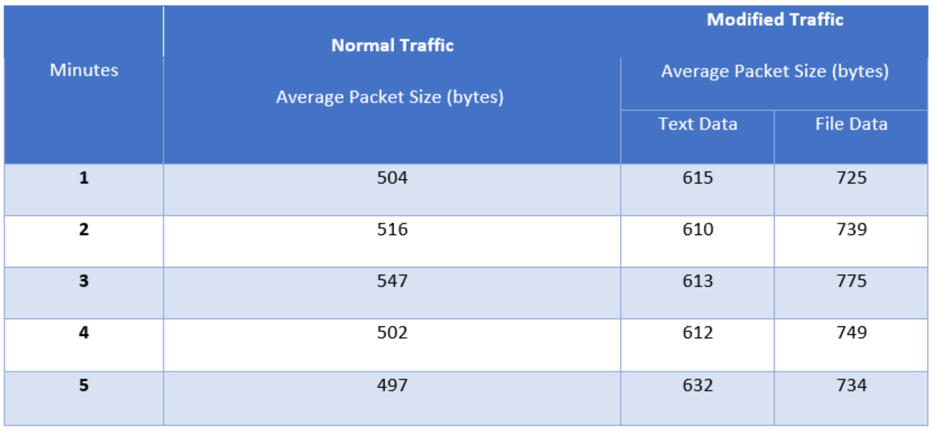


Figure 6.3: Network Traffic Difference for All Audio Streams

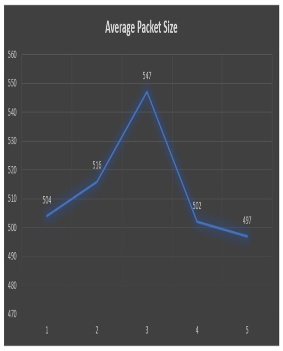
6.2 Video Stream Results

As we experimented with audio stream for five minutes of sample rate, similarly, we did with video streams. And add both type of data e.g. encrypted text, and file stream etc.

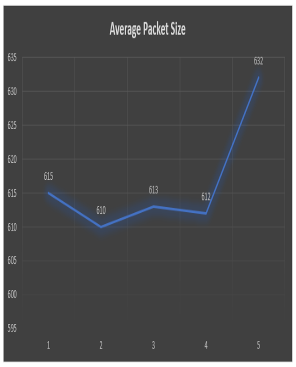


Minutes	Normal Traffic	Modified Traffic	
	Average Packet Size (bytes)	Average Packet Size (bytes)	
		Text Data	File Data
1	504	615	725
2	516	610	739
3	547	613	775
4	502	612	749
5	497	632	734

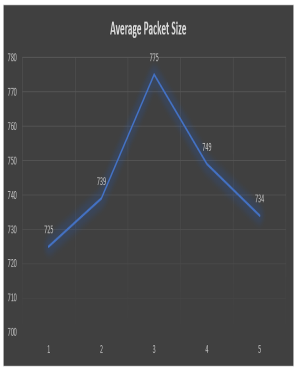
Figure 6.4: Average Packet Size of all Video Streams



(a) Normal Traffic



(b) With Text Data



(c) With File Stream

Figure 6.5: Network Traffic for Video Streams

Figure 6.5 shows that after adding additional data average size of packet of normal traffic is 513 bytes, that rose by adding text data to 616 bytes, and with file stream it reached to 744 bytes on average.

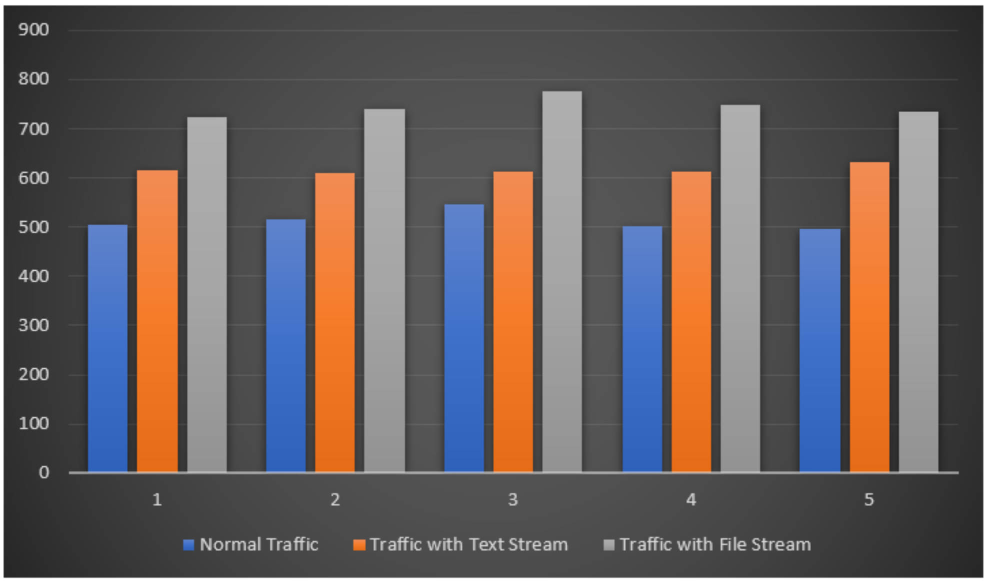
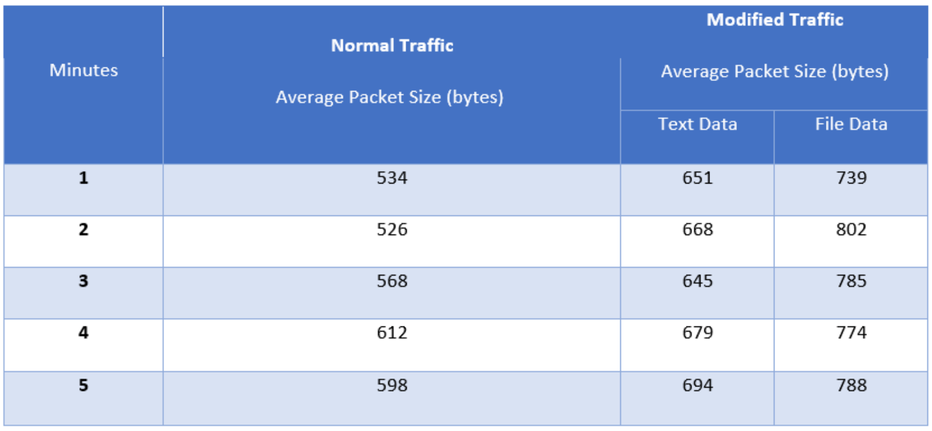


Figure 6.6: Network Traffic Difference for All Video Streams

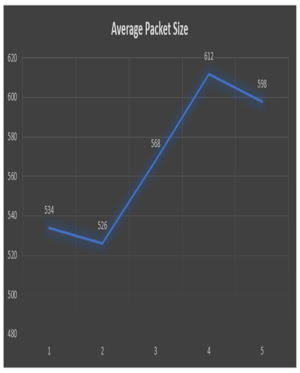
6.3 Screen Sharing Results

As discussed in previous sections, with 5 minutes of sampling rate captured traffic of screen sharing stream with WebRTC using Wireshark. And we got following results:

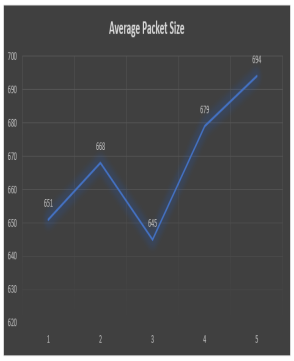


Minutes	Normal Traffic	Modified Traffic	
	Average Packet Size (bytes)	Average Packet Size (bytes)	
		Text Data	File Data
1	534	651	739
2	526	668	802
3	568	645	785
4	612	679	774
5	598	694	788

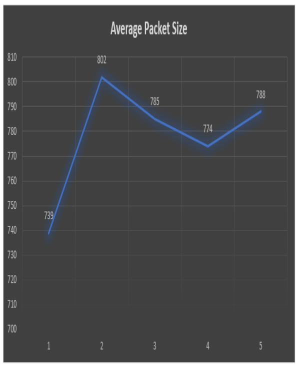
Figure 6.7: Average Packet Size of all Screen Sharing Stream



(a) Normal Traffic



(b) With Text Data



(c) With File Stream

Figure 6.8: Network Traffic for Screen Sharing Streams

Above given table and graphs are showing that how traffic profile is modified by adding some additional data in data channel created with RTC peer connection. Average size of packet in normal traffic rose from 568 bytes to 668 bytes by adding some encrypted text data and reached to 778 bytes with file stream.

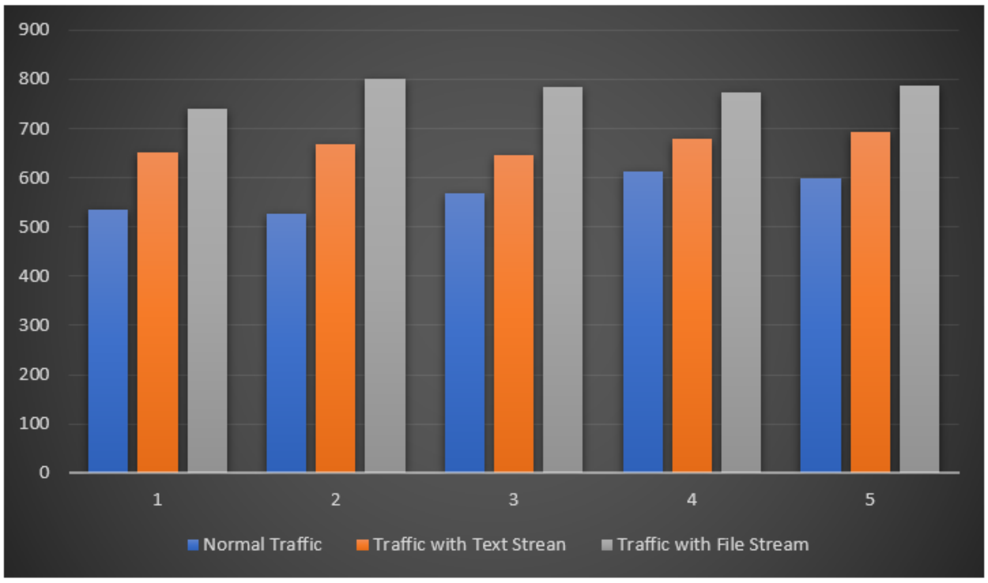


Figure 6.9: Network Traffic Difference for All Screen Sharing Streams

Chapter 7

Conclusion

Network traffic analysis plays key role in network management and network traffic flow. Network detection and extracting packet statistical information can cause privacy and confidentiality issues. Even encrypted traffic also contains sensitive information in packet header that can also cause to detect or even classify network traffic. We modified network traffic profile by adding some additional encrypted data to fool detectors. We established peer-to-peer (P2P) connection using WebRTC and tested on Video Chat, Audio Chat, and Screen Sharing. We modified these streams by adding additional encrypted using data channel that doesn't affect on live communication. This additional data modify packet size that changes traffic profile and detectors, censorship authorities will see false traffic profile. This vulnerability of WebRTC can also cause to be attacked by different attackers e.g. DoS Attack etc. This research contributes in maintaining privacy of a network and to avoid detection and censorship. Hopefully, this will give some new and interesting ideas and aesthetics to research field.

7.1 Future Work

Modifying network traffic profiles can give new aesthetics to research field. We can manipulate different statistical properties of network packets. We can also use silent gaps of a packet to modify its properties that will help us to reshape network traffic and will be difficult to detect and classify. We can also make some algorithmic approach to add WebRTC data. We can check packet size and modify it according to protocol definition. These approaches will help to secure more a network.

Bibliography

[1] Charles V Wright, Lucas Ballard, Scott E Coull, Fabian Monrose, and Gerald M Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 35–49. IEEE, 2008.

[2] R Dingledine, N Mathewson, and P Syverson. Tor: the second-generation onion router in'proceedings of the 13th conference on usenix security symposium-volume 13'. *USENIX Association, San Diego, CA*, page 21, 2004.

[3] Yuzhi Wang, Ping Ji, Borui Ye, Pengjun Wang, Rong Luo, and Huazhong Yang. Gohop: Personal vpn to defend from censorship. In *16th International Conference on Advanced Communication Technology*, pages 27–33. IEEE, 2014.

[4] Andrew M White, Austin R Matthews, Kevin Z Snow, and Fabian Monrose. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In *2011 IEEE Symposium on Security and Privacy*, pages 3–18. IEEE, 2011.

[5] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. Skypemorph: Protocol obfuscation for tor bridges.

In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 97–108, 2012.

[6] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. Stegotorus: a camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 109–120, 2012.

[7] Qiyan Wang, Xun Gong, Giang TK Nguyen, Amir Houmansadr, and Nikita Borisov. Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 121–132, 2012.

[8] John Geddes, Max Schuchard, and Nicholas Hopper. Cover your acks: Pitfalls of covert channel censorship circumvention. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 361–372, 2013.

[9] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*, pages 65–79. IEEE, 2013.

[10] Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing around decoys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 85–96, 2012.

[11] Philipp Winter and Stefan Lindskog. *How the great firewall of china is blocking tor*. USENIX-The Advanced Computing Systems Association, 2012.

[12] Amir Houmansadr, Thomas J Riedl, Nikita Borisov, and Andrew C Singer. I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention. In *NDSS*, 2013.

[13] Sam Burnett, Nick Feamster, and Santosh S Vempala. Chipping away at censorship firewalls with user-generated content. In *USENIX Security Symposium*, pages 463–468. Washington, DC, 2010.

[14] Josh Karlin, Daniel Ellard, Alden W Jackson, Christine E Jones, Greg Lauer, David Mankins, and W Timothy Strayer. Decoy routing: Toward unblockable internet communication. In *FOCI*, 2011.

[15] Shuai Li, Mike Schliep, and Nick Hopper. Facet: Streaming over videoconferencing for censorship circumvention. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 163–172, 2014.

[16] Reham Taher El-Maghraby, Nada Mostafa Abd Elazim, and Ayman M Bahaa-Eldin. A survey on deep packet inspection. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, pages 188–197. IEEE, 2017.

[17] Justin Levandoski. Application layer packet classifier for linux. *http://l7-filter. sourceforge. net/*, 2008.

[18] Silvio Valenti, Dario Rossi, Alberto Dainotti, Antonio Pescapè, Alessandro Finamore, and Marco Mellia. Reviewing traffic classification. In *Data Traffic Monitoring and Analysis*, pages 123–147. Springer, 2013.

[19] Bin Hu and Yi Shen. Machine learning based network traffic classification: a survey. *Journal of Information and Computational science*, 9(11):3161–3170, 2012.

[20] Zigang Cao, Gang Xiong, Yong Zhao, Zhenzhen Li, and Li Guo. A survey on encrypted traffic classification. In *International Conference on Applications and Techniques in Information Security*, pages 73–81. Springer, 2014.

[21] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[22] Colin Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

[23] Sam Dutton et al. Getting started with webrtc. *HTML5 Rocks*, 23, 2012.

[24] Shaojian Fu and Mohammed Atiquzzaman. Sctp: State of the art in research, products, and technical challenges. *IEEE Communications Magazine*, 42(4):64–76, 2004.

[25] Kazuhiro Minami and David Kotz. Secure context-sensitive authorization. *Pervasive and Mobile Computing*, 1(1):123–156, 2005.

[26] Wojciech Mazurczyk, Maciej Karas, and Krzysztof Szczypiorski. Skyde: a skype-based steganographic method. *arXiv preprint arXiv:1301.3632*, 2013.

[27] Wojciech Mazurczyk, Maciej Karaś, Krzysztof Szczypiorski, and Artur Janicki. Youskyde: information hiding for skype video traffic. *Multimedia Tools and Applications*, 75(21):13521–13540, 2016.

[28] Osamah Ibrahiem Abdullaziz, Vik Tor Goh, Huo-Chong Ling, and Kok-Sheik Wong. Network packet payload parity based steganography. In

2013 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (CSUDET), pages 56–59. IEEE, 2013.

[29] Wenxuan Zhou, Amir Houmansadr, Matthew Caesar, and Nikita Borisov. Sweet: Serving the web by exploiting email tunnels. *arXiv preprint arXiv:1211.3191*, 13, 2012.

[30] Sheharbano Khattak, Laurent Simon, and Steven J Murdoch. Systemization of pluggable transports for censorship resistance. *arXiv preprint arXiv:1412.7448*, 2014.

[31] M Anirudh, S Arul Thileeban, and Daniel Jeswin Nallathambi. Use of honeypots for mitigating dos attacks targeted on iot networks. In *2017 International conference on computer, communication and signal processing (ICCCSP)*, pages 1–4. IEEE, 2017.

[32] Rolando Herrero. Encapsulating and tunneling webrtc traffic, March 16 2017. US Patent App. 14/855,542.

[33] Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *SODA*, volume 2, pages 94–103, 2002.

[34] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. Profiliot: a machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*, pages 506–509, 2017.

[35] Hooman Mohajeri Moghaddam. Skypemorph: Protocol obfuscation for censorship resistance. Master's thesis, University of Waterloo, 2013.

[36] Sambuddho Chakravarty, Vinayak Naik, Hrishikesh B Acharya, and Chaitanya Singh Tanwar. Towards practical infrastructure for decoy routing (positional paper). In *Proceedings of the Workshop on Security of Emerging Networking Technologies (SENT) Held in Conjunction with 22nd Network and Distributed System Security (NDSS) Symposium. Internet Society*, 2015.

[37] Tariq Elahi, Colleen M Swanson, and Ian Goldberg. Slipping past the cordon: A systematization of internet censorship resistance. *Centre for Applied Cryptographic Research (CACR), University of Waterloo, Tech. Rep*, 10, 2015.

[38] Stefan Köpsell and Ulf Hillig. How to achieve blocking resistance for existing systems enabling anonymous web surfing. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 47–58, 2004.

[39] Hyunwoo Yu, Eunsu Lee, and Suk-Bok Lee. Symbiosis: Anti-censorship and anonymous web-browsing ecosystem. *IEEE Access*, 4:3547–3556, 2016.

[40] Sergey Frolov and Eric Wustrow. The use of tls in censorship circumvention. In *NDSS*, 2019.