ETHIX : HYBRID VPN

Thesis supervisors

promotor: Prof. dr. M. Conti
copromotor: dr. A. Giaretta

Thesis Committee:

Rector Magnificus,	chair
Prof. dr. M. Conti,	Technische Universiteit Delft

Independent members:

Prof. dr. F. A. Kuipers	Technische Universiteit Delft
Prof.dr. A. Zarras	Technische Universiteit Delft

Prof. dr. M. Conti has contributed to this dissertation.
An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

SUMMARY

The world is getting more and more censored, and the currently available solutions work but with high latency costs. We proposed Ethix, a low latency general-purpose architecture with differential privacy, low network overhead and high theoretical censorship evasion. The Ethix system is a hybrid VPN architecture that uses both centrally controlled components as well as an organized peer-to-peer network. This paper lays out the design introducing a session mixing framework which allows for differential privacy from the proxying server. This differential privacy results in low-quality logs as the data to identity and link cannot be established. Evaluating the Ethix system is done by implementing it using python and amazon ec2 instances and evaluating Ethix against various Adversarial models. The key results indicate that the system in a non-optimized non-deployed setting can reach speeds up to 16 Mpbs, with a minor increase in latency at 5ms for all packet processing. Lastly, this paper argues that the system proposed provides higher privacy than traditional VPNs and is resistant against malicious clients, honest but curious proxy servers, censoring servers and global passive adversaries.

PREFACE

This thesis was a learning experience that challenged me from time management to emotional management. I am grateful that I was allowed to do my design and research. From the start of this project, doctor Alberto Giaretta and Professor Mauro Conti have guided me and given me their time and effort. I would also like to thank Professor Claudia Diaz for taking the time to meet with me. In general, I am grateful for the research being conducted in privacy-preserving technologies, especially the Tor, VPN gate and The Nym tech teams. You make me want to be better. I want to thank Shubham Khandhar for being there every step of the way and asking questions that I was afraid to ask myself.

1

INTRODUCTION

Russia has banned 6 [1][2] VPN services in 2021 and plans to ban more. China has already achieved stunning success in blocking existing VPNs, and India has tested their VPN blocking system in 2019 [3][4]. These events are not isolated: countries are implementing more restrictive censorship measures from Iran to Australia. More restrictive censorship has resulted in a market for censorship evasion software. Commercial VPN services comprise the majority of the market. Commercial VPNs are easy to use and efficient but increasingly easy to block.

Globally, internet censorship is on the rise in the number of countries restricting internet access and the number of websites blocked. The rise of internet VPN censorship now encompasses more than 3 billion people, with China and India accounting for 2.6 billion of them. This has resulted in a significant need for censorship circumvention applications. This has resulted in countries such as China, Russia, and India targeting applications such as VPNs and other censorship circumvention tools. While the arms race continues between censors and VPNs, censors are winning with 76% of VPN services blocked in China.

Other technologies circumventing censorship are available, but they incur high latency and download speed overhead [5]. These performance decreases make systems like Tor and Nym unattractive for regular users. Furthermore, these systems allow untrustworthy users to analyse the traffic, leading to information leakage and critical vulnerabilities [6].

1.1. CONTRIBUTION

The contribution of this paper is a system that can provide higher censorship resistance and privacy properties while minimally increasing latency and download speed overhead. We propose a novel architecture, routing system and packet formats to allow regular users to access the internet freely.

1

1.2. STRUCTURE

The remainder of this thesis is structured as follows: Section Chapter 2 covers essential background knowledge, such as what censorship currently looks like, what properties are essential to measuring privacy, and how latency and download speed scale for VPN systems. Chapter 3, using the background knowledge of the previous section, lays out the requirements of Ethix, the system proposed in this thesis. Chapter 4 covers in-depth the design of Ethix, discussing the components and the different interactions between such components. We discuss and describe the implementation of the design in Chapter 5. Chapter 6 evaluates the proposed architecture in terms of the criteria previously covered in Chapter 3, and our results are compared to the state-of-the-art in Chapter 7. Finally, Chapter 8 discuss future work and limitations of this thesis.

2

BACKGROUND

The background chapter provides context for the evaluation criteria of the proposed system. The system proposed aims to be evaluated on four aspects: censorship resistance, privacy, latency and download speed. For each of these aspects, a section is dedicated to this chapter. Each section aims to communicate the factors that indicate whether a system has an aspect and briefly introduce the background context of said aspect.

2.1. STATE OF CENSORSHIP

This section covers what censorship is, the current implementations of censorship around the globe, identify standard censorship techniques used by censors, and evaluate how effective the standard censorship techniques are.

Censorship is the suppression of speech, public communication, or other information because such material is considered objectionable, harmful, sensitive, or inconvenient. Censorship can be conducted by governments, private institutions, and other controlling bodies. We examine political and religious censorship further as these are generally the motivators for internet censorship, as evidenced by Iranian, Chinese, Russian and Syrian censorship implementation. Political and religious censorship takes numerous different shapes, but the most common are restrictions of free speech against the ruling party and the selected religion of the country. Censorship is not used to prevent all negative information from being accessible but sufficient to sway public opinion. This goal means censorship is to censor enough to ensure that the average voter or state member does not have sufficient knowledge to disagree or connect and collaborate with others.

2.1.1. IMPLEMENTATIONS OF CENSORSHIP

Censorship is implemented in different ways across different countries. The following countries use political censorship in novel, unique or grand ways and act as diverse implementations. While other implementations exist since China has collaborated with multiple countries to strengthen their censorship systems, more and more countries are

using China's sophisticated internet censorship methods.

CENSORSHIP IN CHINA

The People's republic of China has deployed the largest and most comprehensive censorship system. The Great Chinese firewall (GFW) uses network black-holing, DNS spoofing, Url filtering, DPI QOS filtering, TCP reset and MITM TLS. Furthermore, China uses active probing such that once a legitimate client talks to a Tor/VPN server, there is an unsolicited TCP/IP connection that follows [7]. This TCP/IP connection probes the service to see whether it is a Tor/VPN node and if the handshake completes, the IP is blocked by GFW. In total, China aims to solidify its political position by blocking more than 8,000 websites, including Wikipedia, Google and most websites that encourage anti-unified-state thinking.

CENSORSHIP IN IRAN

Iran has a solid Muslim base, and this has been reflected in their internet censorship banning any material which speaks out against the religion. The primary technique that is used to achieve this is a blocklist of 15,000 websites. Furthermore, they are unique in routing all communication through a state-controlled server making Deep packet inspection (DPI) possible and highly effective. Iran detects TLS-VPNs and uses TCP-RST packets to try to stop them. Not only are the censorship technologies used in Iran, but it is also illegal to use encryption as it would infringe on Article 10 [8] which effectively prohibits internet users and companies from using encryption, or protecting data, in a manner that would "deny access of authorized individuals to data, computer and telecommunication systems."

CENSORSHIP IN RUSSIA

Russia aims to prevent criticism against state members; it does this by employing laws banning VPNs and circumvention technology and a firewall built with the help of architects from the GFW. Russia currently uses DPI and blocklists to restrict prohibited content. Furthermore, Russia stores all data for six months after it is sent.

CENSORSHIP IN SYRIA

Syria has been trying to stifle criticism of its political regime, religion via censorship. Unlike the other regimes mentioned in 2017, Syria unblocked several anti-Syria websites, but sadly, while they have moved further and further away from technical censorship, they have moved to physically assert dominance the same year that websites were unblocked 12 citizen journalists were killed.

2.1.2. CENSORSHIP TECHNIQUES

The censorship techniques used depends on the technologies used to request information. Some technologies aim to circumvent censorship, such as proxying, hiding IPs, and obfuscating traffic. These technologies have been adapted to by more sophisticated censorship firewalls such as the GFW. Lastly, we cover what level of censorship avoidance different systems produce.

Censorship is conducted using control points. These control points include internet backbone servers, internet exchange points (IPX), satellite access points, and optical

fibre links between continents. These servers do provide censors access within the country. Further control points include internet service providers (ISPs), which can provide nation-states to monitor traffic within the country. Individual institutions can further be controlled to restrict access to certain content or install spyware on their system.

PLAIN-TEXT TRAFFIC

In this section, we describe censorship techniques that are used to censor traffic sent in plain text. Servers use keywords to censor plain text traffic by searching for non allowed words like "VPN" or monitor website requests using DNS or URLs to prevent access to websites like Google. This section covers three plain text censorship techniques, namely DNS filtering, URL filtering, and connection resets. These techniques are helpful for censors because they can prevent users from gaining more sophisticated censorship-resistant technologies. DNS filtering hijacks the Domain Names System (DNS) drop and modifies traffic requesting blocked websites. URL filtering scans URLs for censored keywords, and if packets match non allowed keywords, they are dropped. URL filtering only affects the HTTP protocol but is effective against non-technical users as generally, their experience is not sufficient to use other protocols. This technique was used to filter for requests to VPN websites, effectively making it harder to access technologies that would circumvent censorship. Lastly, for plain-text traffic, TCP connections can be scanned for unwanted/censored material, and if it is detected, the TCP connection is reset using a TCP reset packet. Receiving TCP reset packets results in the client dropping the connection.

PROXIED TRAFFIC

This section describes the censorship techniques used against proxied traffic. To prevent plain-text traffic censorship, users proxied traffic over VPN servers or their servers. Censors developed new censorship techniques in order to prevent this proxying of traffic. These techniques target proxy servers (IP enumeration and blocking) or proxy packets (Protocol fingerprinting). IP enumeration attacks can be used to construct lists of IPs that are put on a block list. These attacks use publicly available data, for instance, IPs provided by a VPN service required for their users to use the system. Not only this but VPN services are generally located in similar IP blocks owned by cloud providers [9] such as AWS and Azure, which allows blocking a large number of VPN servers with minor unwanted damage. Attacking proxy packets can be done by the use of protocol fingerprinting. Standard VPN protocols such as OpenVPN, which multiple VPN providers use, can be fingerprinted, and the traffic is identified. If packets can be identified as part of proxy connections, these packets can be dropped, thus censoring the traffic.

HIDDEN SERVICES AND OBFUSCATED TRAFFIC

To prevent IP enumeration, Tor has developed hidden services that prevent excessive sharing of network participants and, as such, make network enumeration more difficult. Scramble suit and OBFS4 [10], [11] can be used to obfuscate traffic such that it is more difficult to fingerprint protocols. These protocols make the traffic look like encrypted data, which is difficult to fingerprint. This section describes the censorship techniques used to censor traffic that uses hidden services or obfuscated traffic. These techniques are IP active probing, Quality of Service (QoS) filtering, portal censorship, and network

reconnection. Portal censorship and network reconnection are not currently fixable by any system and are not addressed in this thesis.

IP active probing was a method initially developed to find Tor hidden services. The attack works by monitoring suspicious traffic going over, censoring servers and then querying the server part of the suspicious traffic and pretending to be part of the network. If the server responds, it is validated as a server of an unwanted censorship circumvention system, and the IP is put on a block list. There are several methods suggested to prevent this type of attack, namely a salmon ladder [12] or the hidden services working together to identify probing servers to blocklist their IPs on a system level [13]. QoS filtering is the most sophisticated currently in use censorship system available. It can "learn, filter and block" using connection monitoring and DPI (deep packet inspection). It uses these tools to gather information and assign connections suspiciousness scores. Then depending on how suspicious a connection is, a percentage of the traffic is blocked. This system is variable such that during times where a government wants to suppress dissent, such as during a national embarrassment actively, the percentage drop rate can be increased [14]. This is an effective way to censor encrypted traffic as the entropy of packets can be measured. This attack is currently not effective against pluggable transports such as obfs4. Portal censorship is the act of applying pressure to information sharing indexing platforms such as Google or Bing in order for them to delist information. Portal censorship bypasses any censorship circumvention method mentioned previously and would be difficult to detect. Although portal censorship requires company participation, large nation-states such as China represent an enormous market, and thus incentive exists for companies to comply with requests to delist such information. This can occur on only on a search engine basis but also on app stores [15]. An example of this would be the 2021 blocking of the smart voting app from app stores due to Russia applying pressure to Google and Apple. Lastly, if a user is deemed sufficiently vocal or anti-agenda, their network connection can be disconnected. Since Internet service providers (ISP) are generally state-controlled entities, users can be blocked from using the internet.

2.1.3. STATE OF CENSORSHIP EVASION

This section aims to give an accurate picture of the current efficacy of different censorship evasion methods available to users. Since censorship techniques are not perfectly accurate, technologies that are theoretically censor-able are still helpful.

NO SYSTEM USED

If no system is used to avoid censorship, the level of censorship experienced will depend strongly on the originator country of the request. If the countries censorship technologies are sufficiently sophisticated, this will result in a close to perfect censoring of information.

VPNS

If a commercial VPN is used, users can circumvent most censorship in countries without sophisticated censorship firewalls, but in countries like China, many VPNs struggle as DPI (Deep packet inspection) is used to identify VPN packets. Furthermore, sophisticated censorship firewalls develop dynamic blocklists which enumerate VPN server IPs

and put them on a blocklist. DPI can still be circumvented by several VPN providers such as Express VPN, NordVPN and others. The circumvention method is in 2021 relies on the use of traffic obfuscators such as obfs4 or relying on a protocol that is not standard or not fingerprinted by the blocking censorship firewall. While IP enumeration is harder to prevent, the large number of servers owned by services such as Express VPN makes it difficult to block all systems consistently. The current state of censorship avoidance with VPNs is such that only 76% of providers can bypass blocking while an estimated 98% of all VPN servers are down [16]. This VPN block rate means that censors are highly effective at censoring even with the use of VPNs.

ANTI-CENSORSHIP SYSTEM CENSORSHIP AVOIDANCE

Other systems such as VPN-gate, Tor and Sentinel are theoretically more resistant against censorship than commercial VPNs, but several attacks against these systems are available to censor them. Tor has been at the forefront of the censorship wars for 15 years with China, and attacks have been developed to reduce its use. Techniques such as active probing have proved effective for a long time, although systems such as snowflake [17] could end this. Furthermore, newly deployed systems such as the Nym network and orchid mentioned in the related work section, which the GFW has not had time to adjust to, could provide long term solutions for users.

2.2. STATE OF PRIVACY

In this section, we cover what privacy is, which attackers are privacy threats and evaluate how effective current privacy-enhancing software is against the attackers in providing privacy to the user.

Privacy is the ability of an individual or group to seclude themselves or information about themselves. There are different forms of privacy, but the most relevant for internet usage on a network level are expressed in terms of anonymity, unlinkability, undetectability, and unobservability according to the IETF guidelines [18].

Privacy component definitions:

Anonymity definition: "Anonymity of a subject means that the subject is not identifiable within a set of subjects, the anonymity set."

Unlinkability definition: "Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot sufficiently distinguish whether these IOIs are related or not"

Undetectability definition: "Undetectability of an item of interest (IOI) from an attacker's perspective means that the attacker cannot sufficiently distinguish whether it exists or not"

Unobservability definition: "Unobservability of an item of interest (IOI) means undetectability of the IOI against all subjects uninvolved in it and anonymity of the subject(s) involved in the IOI even against the other subject(s) involved in"

2.2.1. ATTACKERS AND RESOURCES

Any system which can gain information about the contents of messages, the legitimacy of a message, or senders/receivers of a message is considered an attacker. Most privacy systems need to be private against Malicious users, exit proxies, control points, and global passive adversaries (GPAs).

MALICIOUS USERS

Other attackers can introduce new users into systems such as Tor or Ethix and then can gather information. In the case of systems such as Tor, if sufficient numbers of fake users are introduced, they can monitor traffic flows to determine from and who packets are going to.

EXIT PROXIES

Exit Proxies provide the services of releasing packets into the internet out of the system. This can allow them to inject, read and drop packets. This has resulted in numerous attacks against Tor and other systems that use unprivileged, unverified services as exit proxies. In the case of VPNs, exit proxies are the VPN server.

CONTROL POINTS

All data going over the internet requires control points (defined in the state of censorship section). These internet control points can inject, read and drop packets. Although these packets are encrypted to inject or read, the control points need a way to man in the middle (MITM) attack.

GPAS

Global passive adversaries are state-run organizations such as the NSA. Their goal is to gain as much information as possible. This thesis assumes they have sufficient resources to monitor all packet flows between any two nodes.

2.2.2. STATE OF PRIVACY SYSTEMS

In this section, we identify a variety of systems that are used to ensure user privacy and evaluate them in terms of privacy laid out in section 2.2 against attackers laid out in section 2.2.1.

NO SYSTEM USED

When no anonymity system is used, the attacker's malicious users and exit proxy do not exist. So the privacy of the system is evaluated against control points and GPA. The result of these evaluations is that when no system is used, the privacy of the system meets none of the privacy terminology laid out in section 2.2. Control points can monitor sessions, and there is no proxying going on, so both the source and destination will be revealed. Thus there is no sender or receiver anonymity. Further, the items of interest (IOI) (Packets, Sessions) can be linked to a specific IP:port combination, and thus, it is possible to link multiple IOIs. Further, since the packet is observed, it can be concluded that it exists and is most likely not fake, and as such, the system would have no undetectability and by reasoning [18]also no unobservability. Since GPAs can monitor all connections, they can break all privacy similarly to the control points.

VPNS

When traffic is proxied over a server like a commercial VPN, three attackers can gain information. These three attackers are exit proxies, internet control points and GPAs.

Exit proxies can modify, read and drop packets. Furthermore, they can identify the sender and receiver of packets and thus can break all privacy guarantees. The internet control points can no longer see the receiver of information, e.g. the requested website. Further, a stronger unlinkability exists as control points can no longer differentiate multiple different sessions by unique IP:port combinations. Proxied traffic still does not meet the requirements of undetectability because no false packets are created, and thus unobservability is further not guaranteed. The GPAs can use timing analysis to view all traffic flows and thus can break all privacy definitions.

PRIVACY SYSTEM PRIVACY GUARANTEES

The more complex systems generally include higher privacy guarantees, but the exact properties of the systems depend on the implementation. Systems like Tor provide high levels of anonymity while providing unlinkability on a per use (per browser session being opened) basis.

2.3. LATENCY CONCEPTS

Latency is the time taken between two events in the case of computer networks. It is generally measured using round trip delay (RTD), which measures the time between packet request and packet response. Latency must be low for several applications such as voice calls and gaming. These applications fail to function correctly with a latency of greater than 150 ms [19]. Since all traffic travels through the privacy/anti-censorship system, the overhead latency must be kept sufficiently low. The latency overhead of the system is influenced by three factors distance travelled, the computation required and the number of proxies.

2.3.1. DISTANCE

The distance is the major component of latency, with the minimum RTD around the world (France-New Zealand) being around 300 ms. 300 ms is due to the physical limitations of the components with the speed of light in a non-vacuum (glass fibre cables). The speed of light in a non-vacuum combined with processing along the way makes distance the most significant component of latency.

2.3.2. COMPUTATION

The time is taken to compute, decrypt a packet is negligible in terms of latency. The implemented system takes around 1ms per request, which is standard compared to other anonymity systems.

2.3.3. NUMBER OF PROXIES

Furthermore, If multiple different proxies are used, as is the case in Tor and Nym networks, the distance can add up multiplicative with the number of nodes chosen randomly on the globe. In the case of Tor, on average, three proxies are taken to and from

the destination, increasing the distance travelled threefold.

2.4. THROUGHPUT CONCEPTS

Throughput is the size of the traffic that can be moved over a system. This thesis is measured in Mbps and determines how quickly a client can download a file. Similarly to latency, it depends on the distance, the computation required per packet, and the number of proxies.

2.4.1. DISTANCE

Unlike latency, distance factors only minorly into download speed but over long distances, a download speed decrease is expected as a server along the way is more likely to act as a bottle net as it could be in high demand or does not have the computational power necessary.

2.4.2. COMPUTATION

Computation is the most critical factor for download speed, as the number of messages that can be sent and received depends on the number of packets that can be computed within a second. If a packet takes 1ms of processing time to decrypt and re-encrypt, 1000 packets can be sent a second.

2.4.3. NUMBER OF PROXIES

The number of proxies does not directly determine download speed, but a higher number of proxies increases the chance that one will have insufficient resources and thus act as a bottleneck slowing the entire system down.

2.4.4. EXPECTED DOWNLOAD SPEED

With VPNs hardly throttling internet speeds (depending on the implementation), while anonymity systems such as I2P can transfer at around 0.06 Mbps [20] and Tor being able to transfer at 8.3 Mbps [21].

3

REQUIREMENT ANALYSIS

In this section, we lay out the requirements that we have for the designed system. As outlined in the background and introduction sections, there is a need for low latency systems that allow users to have higher privacy and censorship evasion possibilities.

The niche for the system being designed is thus to have better privacy and censorship evasion properties than VPNs while providing a more general (multiple supported protocols) and usable system for everyday users. The more exact requirements of the systems are outlined in the following sections.

3.1. PRIVACY REQUIREMENTS

The requirements are made for the exit proxy as it is the only component with access to the precise text data. If multiple different attackers work together, this could result in a more significant compromise of privacy.

1. Anonymity: A subject sending a message is not identifiable within a set of subjects, the anonymity set. While various systems can make probabilistic guesses using a-priori knowledge, they cannot validate which subject sent or is receiving the message.

2. Unlinkability: sessions (unique IP:port combinations) here, items of interest IOI cannot be raelated to one another. Although using DPI, the exit proxy could make probabilistic guesses on who the sender/receiver is.

3. Undetectability: No attacker can determine whether a message is legitimate or a dummy message.

4. Unobservability: Every system except the user is unable to determine the legitimacy of the IOI (session) and further that the subject of exit proxy is unable to tell the other subject (sender/receiver) of that IOI (session).

3.2. ANTI-CENSORSHIP REQUIREMENTS

The anti-censorship requirements of the system are measured against the capabilities of the attackers. The capabilities of a censoring attacker are outlined in the background section and are generally located at points of control. For each class of attack, we outline the requirements we need of the system.

3.2.1. CLEAR TRAFFIC ATTACKS

Since plain text traffic attacks are mitigated by having a proxy, the system requires complete resistance against these attacks.

3.2.2. IP BLOCKING

IP blocking attacks require data about which IPs belong to the network. In this section, we discuss the required enumeration resistance of the designed system.

1. Public IP enumeration: Strong resistance such that only a small number of nodes can be enumerated that way
2. Private IP enumeration: Moderate resistance such that no more than 50 % of the network can be enumerated
3. Active probing: Fully resistant such that no active probing should be possible

3.2.3. QOS FILTERING

QoS filtering is the most sophisticated method of blocking VPNs. QoS filtering is done on a suspiciousness score, and thus, to be resistant against QoS filtering attacks, traffic needs to look inconspicuous.

1. Protocol fingerprinting: Strong resistance such that it is difficult to determine a link between the protocol and the designed system
2. Packet fingerprinting: Strong resistance against such that it is difficult to link packets to the designed system

3.3. PERFORMANCE REQUIREMENTS

Since the system is aimed at regular users, these requirements are set out for an unoptimized implemented system. It should provide sufficiently low latency options to allow for calls, gaming and other low latency needing applications.

1. Latency: Latency such that it is sufficient for calls and gaming here set at < 150 ms
2. Throughput: Moderate download speed such that normal users can interact with the internet here set at 5Mbps

4

ETHIX

This chapter describes the proposed solution. The system consists of three components clients, ID servers and Proxy servers and is discussed in section 4.1. Next, the section 4.2 explains how clients interact with ID servers and proxy servers while maintaining the privacy and avoiding censorship. Furthermore, this chapter describes the packet format that allows for these client interactions. Lastly, the chapter concludes with an overview of how clients connect and integrate into the network.

4.1. COMPONENTS

The Ethix system consists of 3 components clients, ID servers and Proxy servers. Clients are grouped into groups that support each other. The clients interact with the system in two main steps, as seen in fig. 4.1. The first is the system sign up it provides the client with all necessary information for connecting and interacting with the proxy server. The second step is the client proxying data over the proxy server.

4.1.1. CLIENTS

Clients are the users of the system. For a user to be considered a client, he/she must obtain an ID key from the ID server. Clients, before they have joined a group, are only able to communicate with public clients for proxying traffic to the ID server and with the ID server for obtaining a group and group key (Step 1 in fig. 4.1). Once a client has received a group, they can request traffic through their group and the proxy server. The group is a term for a set of clients who support each other by providing each other with services such as proxying, applying pluggable transports and mixing. Groups act as the anonymity set of a client such that the proxy server is unable to determine which client within the anonymity set sent a requested the proxying of a message—the more clients providing services to each other, the more private the group becomes. Furthermore, the overhead latency reduces the more clients are available because groups can be clustered by relational latency. Lastly, some clients act as public entry nodes for the network and thus, censorship resistance scales with the number of clients.

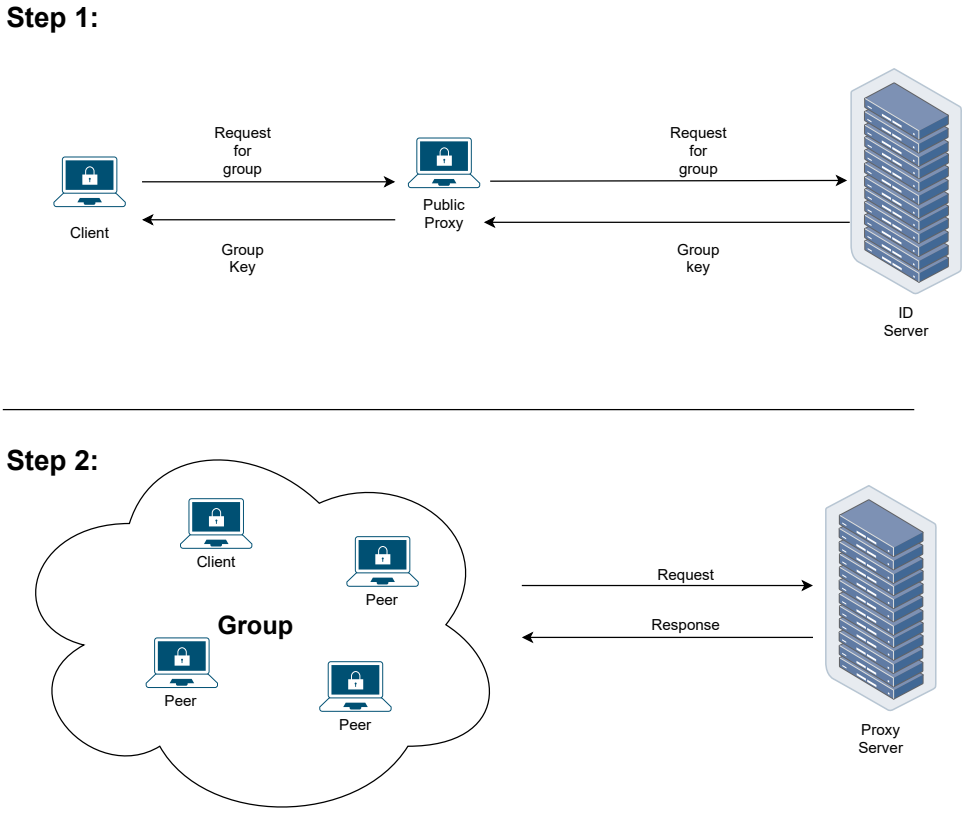

Figure 4.1: Ethix system usage step. In Step 1, a client requests a group to a public proxy which forwards that request to ID server and then provides a group key to the client. In Step 2, a client has joined a group. Requests are proxied through this group and then sent to the proxy server such that the proxy server is unable to determine which group member originated the request. The proxy server responds and the response is routed through the group to the client

4.1.2. PROXY SERVER

The proxy server proxies traffic without knowing who the requesting client is. This is achieved by the sender, not including any information that could track them across sessions (unique IP:Port combinations). To validate that a user is allowed to use the proxy server, the group key is checked. Only participants in the group, as well as the group server, know the group key. This results in the proxy server reducing the possible list of senders to the number of clients within a group. Furthermore, the proxy server is unable to tell who the final recipient is. This is achieved by binding sessions (unique IP and port combinations) to three fields, the next hops IP, an encrypted message for the exit proxy and finally, the symmetry key of the client. These fields are sufficient to return an encrypted data packet to an unknown client. Any incoming packets can be mapped to a session, and thus the three fields allow the proxy server to return it to the sender.

Clients can request updates of knowledge for their known groups from the proxy

server. This allows them to keep accurate information on group participants.

4.1.3. ID SERVER

The ID server is the brain of the system. It stores, deletes, collects and manages system critical information. The core functionality in the ID server is client management, group management, public client databasing, and system security.

CLIENT MANAGMENT

The ID server allows users to sign up to the system to become clients by asking them to complete tasks like filling out a CAPTCHA or sending payment. Once this task is complete, the ID server will issue the requesting party a blindly signed token. The client can use this Blind token in order to charge an ID key with money.

GROUP MANAGMENT

The ID server manages groups sizes such that groups are not under or overpopulated. Furthermore, ID servers can request groups from specific proxy servers to allow for bandwidth management.

PUBLIC CLIENT DATABASING

Subsets of the Ethix system users sign up for their (IP,public_key) to be databased by the ID server. The ID server can hand these clients out as it chooses. These shared IPs can be shared behind a CAPTCHA and linked to IPs to prevent excessive requests significantly enumerating the system. This system has worked for Tor [22].

SYSTEM SECURITY

The ID server prevents network overuse by receiving the data from the proxy server, which can tally the amount of data requested by groups. If the amount of data requested by a group is disproportionate, the ID server can break up the group and re-distribute the participants to various groups. Thus, the ID server will identify probabilistic-ally the overusing client (ID key) and remove them from the system. Furthermore, the ID server protects against active probing of peers by only letting peers respond if the probing system is validated. If a validated client is trying to enumerate the network, the ID server will notice a high number of different requests for validation for a singular ID key. Which the ID server can remove from the system before the key holder can enumerate a significant part of the network

4.2. SESSION MIXING

Once a client is established in the network as covered in section 4.4 the client can interact with the proxy server. Ethix makes this interaction differentially private by the use of session mixing. Session mixing is done by creating a route per new session on a client. Session defined here as a collection of unique IP:Port combinations. Each new session selects a type of route shown in fig. 4.2. The route type determines whether a packet is routed through an entry or exit proxy and the proxy server. The entry and exit proxies are other members of the group that the client signed up to. Since this is done per session, a single client sends and receives packets from various other IPs.

Session mixing results in the proxy server being unable to tell whether the sender or receiver listed on the IP layer of the packet is correct. Since the proxy server always receives the packet in the same format. This can be seen in fig. 4.2. If the proxy server aims to identify user legitimacy, it thus has to use other means of identification such as an ID number. Multiple clients share this ID number in the Ethix system and thus do not identify them individually. Furthermore, other identifiable packet features are stripped and newly generated per session. For each session, this means selecting a route, generating keys, generating a new session identification hash list, selecting route options such as timing delays, adding splits (packets going to multiple IPs), and injecting fake packets. The combination of these results in privacy for clients from adversaries.

In fig. 4.2 four different types of Routes are shown. For each session, a route type is chosen at a rate defined in a configurable. Depending on the route chosen, packets will be proxied by various peers (other group members). The full route both on the servers and back gets routed through a node in the peer-net. The bridge and exit routes only go through a bridging and an exiting peer, respectively. Lastly, the Basic route goes through neither.

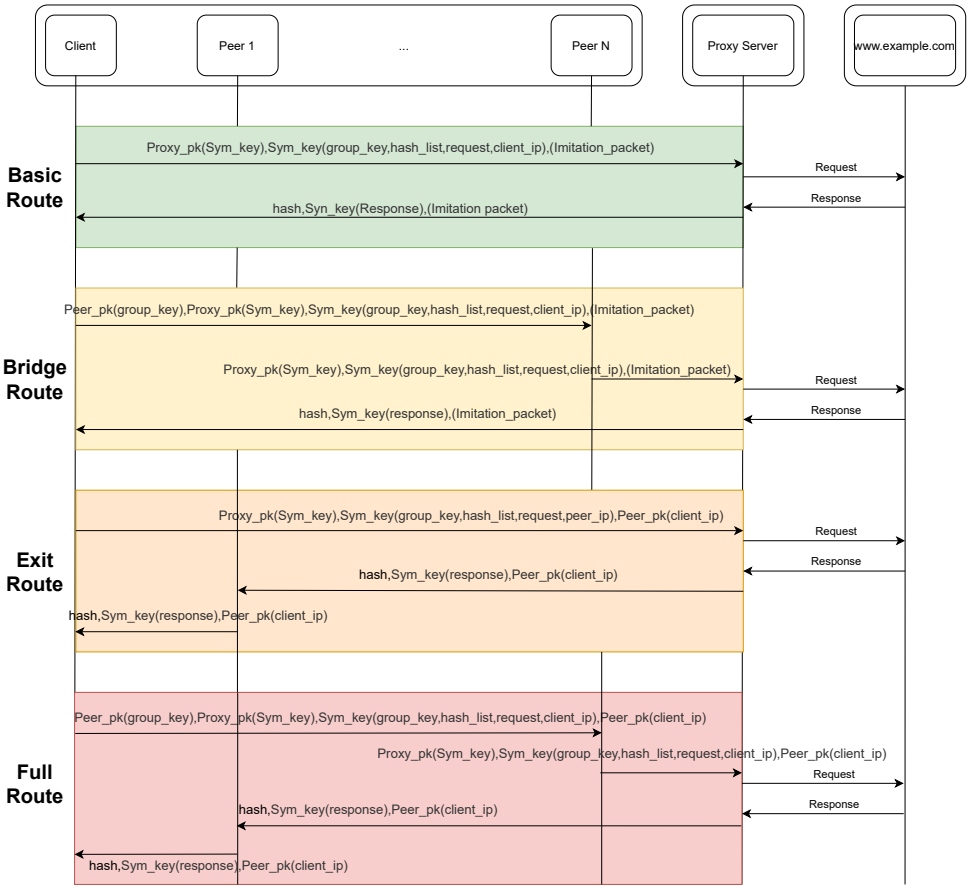

Figure 4.2: Four route types: Basic, Bridge, Exit and Full route types are shown. The basic route in green shows the client sending a packet to the proxy server directly and specifying the client IP as the return address. The bridge route shown in yellow first traverses through a peer in the group. The peer strips the part of the packet intended for it and forwards it to the proxy server. The proxy server returns the response to the request directly back to the client. The exit route is shown in orange, where the client sends its request directly to the proxy server, and the proxy server sends the response back to the peer IP specified in the packet. The peer at the IP specified returns the packet to the client. The full route shown in red is through both a bridge peer and an exit peer. The request is sent first to the bridge and then is forwarded to the proxy server, which sends the response to the request to the exit peer. The exit peer returns the response to the client.

Route name	packet overhead	Benefit	Like route
0: No VPN	0	Low overhead	N/A
1: Basic route	2	Low overhead	4
2: Bridge route	3	Censorship circumvention	3,4
3: Exit route	3	Increases privacy	2,4
4: Full route	4	High Proxy confusion	1,2,3

Table 4.1: Table of different route types, packet overhead (number of additional packets sent and received over the use of No VPN),their benefits, and the routes that look similar from the perspective of the proxy server covered in more detail insection 6.1.2

The different routes have different latency, download speeds, privacy or censorship evasive properties. The more hops through the system are made, the more latency and download speed costs will incur the number of hops can be seen on table 4.1. At the same time, more hops can provide greater privacy and censorship evasion. Furthermore, not all routes look the same from the perspective of the proxy server. If a proxy server is honest and curious and checks whether the sender and receiver are different IPs, it can determine whether the packet is using is not a basic route (if the sender and receiver are different). The routes that look alike can be seen on table table 4.1.

Interestingly the privacy guarantees of the system do not change if all users are only using the basic route type. This is because the server cannot determine which routes through the network are being taken and thus has to treat them all as untrustworthy, which results in the same number of average hops as a regular VPN while providing higher privacy.

Lastly, the routing is randomized further with the use of mixes and false packet modifications. Packets can be mixed such that timing attacks become more difficult using a delay specified within the packet. Ethix-packets can also specify multiple IPs in the next-hop IP to multiply the packet and confuse the sender as to which IP wanted the packet. Lastly, the Ethix-packets can be forged by anyone with a group key. This allows clients to forge packets that could have come from anyone within the system. This is further covered in section 4.3.

4.3. PACKET FORMAT

Packet formats are what make the different routes possible. In fig. 4.3 the seven different packet formats are shown. What is relevant for all seven of these packet formats is the use of timestamps to prevent replay attacks and the use of a list of hashes. These hashes allow clients to determine which symmetric key to use to decrypt the packet. This is accomplished by the proxy server selecting a hash from the hash list and appending it to the front of the encapsulated return packet. The client has a stored mapping between hashes and keys and is thus able to determine which keys to use. The packet, if it is being sent through a bridge, includes a bridge peer encrypted packet. The bridge section can be seen in green in fig. 4.3.

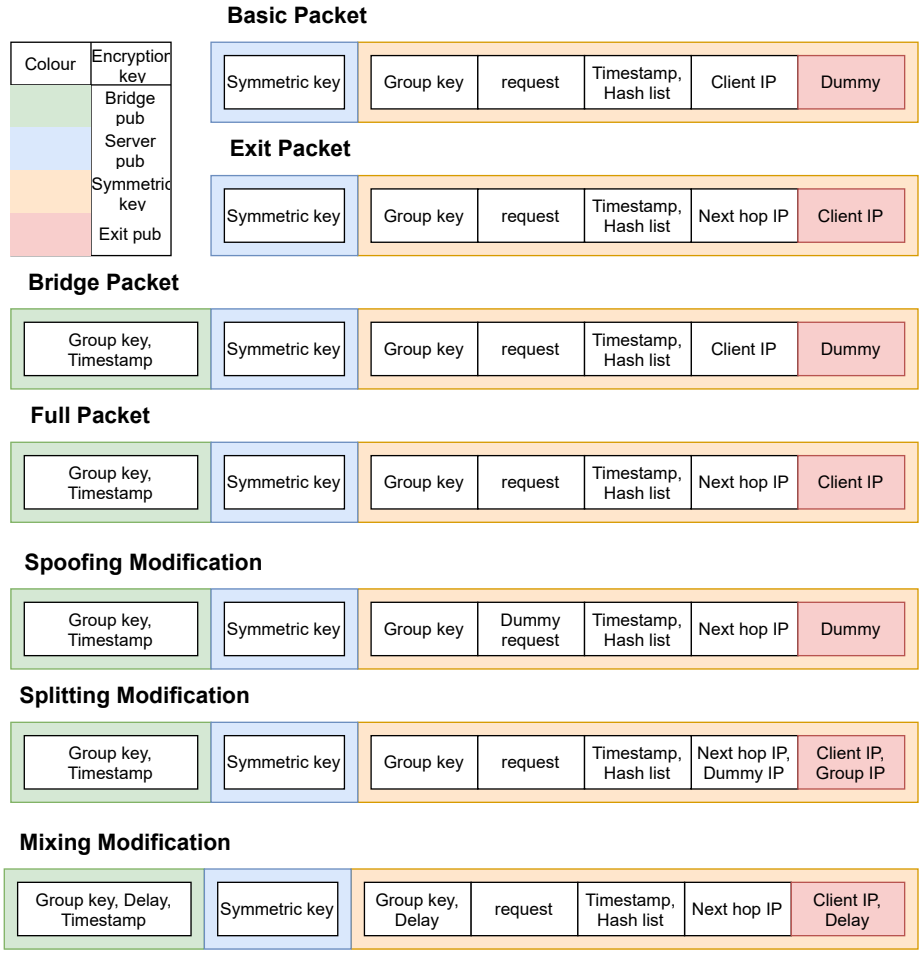

Figure 4.3: The packet format for routes and modification examples for the full route. The basic packet specifies the return address as the client IP. The exit packet specifies an arbitrary next-hop IP and includes the client IP encrypted with the next-hop IPs public key. The bridge packet has an appended section with the group key and time stamp such that the bridge can verify the client. The bridge and full packet are sent to the bridge. The Full packet includes the appended bridge section and specifying the Next-hop IP similarly to the exit packet. The spoofing modification uses a Dummy request but is otherwise identical to other packets. Splitting modifications specify multiple next-hop IPs such that the proxy sends the request to multiple IPs. Lastly, the Mixing modification specifies a delay at any proxy, adding an artificial delay to the packet.

All packets include the server section of the packet. The server section comprises a hybrid cryptosystem where the symmetric key is encapsulated by public-key encryption, and the symmetric key encapsulates the data. This can be seen in the blue, orange and red sections of fig. 4.3. The symmetric key is used for all communication for the session. The data of the server section is further split into two sections the request and the proxy section. The request section contains the validation that a specific client is allowed to make a request. This is in the form of the group key. The request section also contains the request to be executed. The proxy section comprises two components the information for the next hop encrypted by the client and the IP for the server to send it to. Lastly, it contains a list of hashes, which is appended to the return packet.

4.3.1. DATA PACKET TYPES

While the packets have similar features, minor changes to the packet result in different routes taken through the network; the following section discusses all packets that involve no GPA confusion. These data basic packet types are the Basic packet, exit packet, bridge packet, and full packet these can be seen on fig. 4.3.

BASIC PACKET

Basic packets have no interaction with the group, and thus neither the bridge section nor the exit section of the packet is used. Since the proxy server should be unable to determine whether there is a legitimate client that the packet is being sent to, a dummy encrypted exit section is appended at the end of the server packet. This packet would be sent directly to the proxy server, and the proxy server would send all packets from the concurrent session to client IP as can be seen in fig. 4.3.

EXIT PACKET

The exit packet needs to make sure that all packets are routed through an exit peer. This member of the group can be found using the next-hop IP field. This is forwarded to the next-hop IP and then decrypted by the exit peer to reveal the Client IP. This client IP is used to send the packet to the client from the exit peer.

BRIDGE PACKET

Bridge packets are initially sent to the bridge. The bridge validates both the group key and the time stamp. If these are legitimate, the packet is forwarded to the proxy server with the bridge section stripped. The section is stripped to prevent the proxy server from determining whether it came from a bridge. The rest of the packet route is identical to the basic packet route.

FULL PACKET

Full packets route the packet through both a bridge and an exit node. This packet is sent to the bridge peer and validated. Once validated, it is forwarded to the proxy server. The proxy server sends all responses to this packet session to the exit peer. The exit peer decrypts the client IP and sends it on to the client.

4.3.2. GPA CONFUSION MODIFICATIONS

The GPA can monitor all connections between any two nodes within the system. This allows the GPA to perform timing attacks. The spoofing, Splitting, and Mixing modification discussed in this section aim to confuse the GPA and provide privacy. The modifications modify the packets such that some healthy behaviour is exhibited. These modifications generally incur some overhead costs. In fig. 4.3 all three of the GPA confusion modification are shown in the full route packet format but could be utilized in all other route types as well.

SPOOFING MODIFICATION

The spoofing modification modifies a newly generated packet by generating a fake request instead of a legitimate one. These fake requests provide the undetectable property as they act legitimately. This increases confusion for all proxies as any packets could be fake.

SPLITTING MODIFICATION

The splitting modification changes packets such that multiple next destinations are specified. This prevents the sender from knowing who the final recipient is and confuses the GPA since the timing analysis does not indicate who the final recipient is.

MIXING MODIFICATION

The mixing modification allows for the specification of a delay. This delay can be specified at any proxy (proxy server, group peer), which adds an artificial delay to the packet. This reduces the ability of a GPA to perform timing analysis on specific packet flows.

4.4. SYSTEM SIGN UP AND INTEGRATION

This section describes the signup process for clients such that they can use the system as a VPN. This is broken down into three stages of network integration.

4.4.1. INITAL SETUP

Unprivileged is the starting stage for all users. This stage aims to get an ID key without the ID server linking the payment method and payment id with the resulting ID key. This is not shown in this thesis but could be implemented using zero-knowledge proofs [23].

4.4.2. NETWORK MEMBER

The starting state for this stage is that a client has contacted the ID server, and the ID server has replied with the application code and a set of peers and corresponding keys providing bridge services.

4.4.3. NETWORK USER

Once a client has received an ID key, he/she can start on the handshake shown in fig. 4.4. In step 1, the client identifies and proves knowledge of the ID key. The ID server provides the correct proxy server/ group to the client because most of the proxy servers do not need to be publicly revealed and are thus harder to block. Furthermore, this prevents

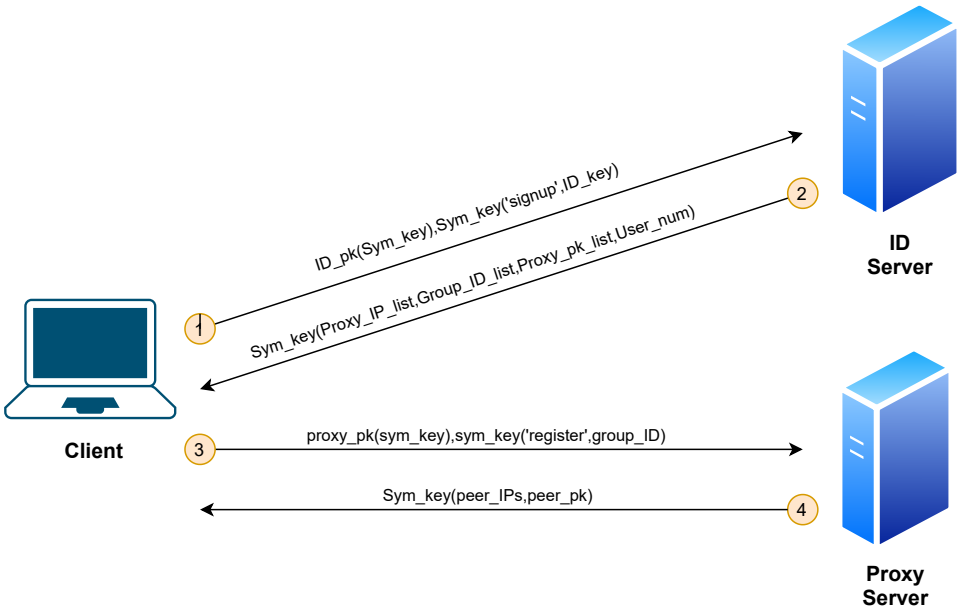

Figure 4.4: Ethix sign up and integration. Step 1: the client requests a sign up to the proxy server with its attached ID key. The ID server validates that the ID key belongs to an allowed user and, in Step 2: responds with the list of proxy IPs with the associated public keys. Furthermore, the ID server provides the client with the number of users such that a client can determine whether the proxy server has added any users. In step 3: the client registers at the group server with the group ID provided by the ID server. If the ID is legitimate, the Proxy server responds in Step 4 with a list of group members.

malicious peers working together with a compromised proxy server from being able to single out specific users as they will be provided with different proxy servers and groups. The ID server will reply in step 2 with a list of groups with the specified parameters from the client. This information is then double-checked with the information gathered in steps 3 and 4 to determine whether the proxy server is working in conjunction with peers. In step 4, the client receives peers whom he/she shares a group with and thus can use to talk to the proxy server.

5

IMPLEMENTATION

The implementation of the Ethix system was made in python3 in approximately 2.5 thousand lines of code. This chapter covers the basic building blocks of the system that enable it to function and provide pseudo-code for ID server, Proxy server, and client.

5.1. TECHNICAL BUILDING BLOCKS

The Technical building blocks section contain the most critical building blocks for the basic infrastructure of Ethix.

5.1.1. TUN/TAP

TUN/TAP devices are virtual network interfaces (VIC) on UNIX-based operating systems. In UNIX operating systems, user-space programs cannot access incoming and outgoing raw Ethernet/IP packets. TUN/TAP devices are designed to fix this problem by providing a network interface monitored/written to via a user-space file descriptor. Physical network interfaces put data "on the wire" while virtual network interfaces write data to a file. In Ethix, TUN devices are used in conjunction with UNIX routing commands to capture all packets leaving the device at an IP level into a file. This file can be monitored using a select statement, and then packets exiting the system can be encapsulated.

5.1.2. PACKET ENCAPSULATION

How a packet is encapsulated depends on the system and its job. What remains constant for both client and server encapsulation is the use of the ECC 25519 public-private cryptography system and the symmetric cryptography system of XSalsa20. These systems are implemented in the python library of NACL boxes which is reliable and well maintained. Both of the cryptography systems are the current state-of-the-art high-speed encryption taking around 0.1ms per full encryption decryption cycle for both symmetric and public-key encryption. 0.1ms was calculated by timing 10000 encryption and decryption steps of a variable and dividing the resulting time by 10000.

CLIENT ENCAPSULATION

Clients encapsulate packets differently depending on which route was chosen through the network. The encapsulating method to generate a full route packet is shown below. The pseudo code shows in line 7 - 9 the hybrid encryption that generates the encrypted server section. Line 4 and 12 show the public key encryption of the bridge and exit component respectively. Lastly the encrypted packet is a concatenation of these three sections. The encapsulation methods for basic, bride, and exit packets have a similar format and can be found in appendix A.

Algorithm 1 Full Route Encapsulation

Input: message,server_pubkey,bridge_pubkey,exit_pubkey,sym_key
Output: encrypted_packet

```
1:  bridge_message,main_message,exit_message = generate_content(message)
2:
3:                                                    ▷ Encrypt bridge message
4:  encrypted_bridge_message = ECC25519(bridge_pubkey,bridge_message)
5:
6:                                                    ▷ Encrypt main message
7:  encrypted_sym_key = ECC25519(server_pubkey,sym_key)
8:  encrypted_message = XSalsa20(sym_key,main_message)
9:  encrypted_main = encrypted_sym_key + encrypted_message
10:
11:                                                   ▷ Encrypt exit message
12: encrypted_exit_message = ECC25519(exit_pubkey,exit_message)
13:
14:                                                   ▷ combine components and return
15: encrypted_packet = encrypted_bridge_message + encrypted_main + encrypted_exit_message
```

5.1.3. SOCKETS

Sockets are used to send encapsulated packets to the other network participants (Servers, peers). For this, python3 UDP sockets were used. UDP sockets were used as this prevents the TCP stacking error [24]. Servers have multiple open ports 6666 for data and 5000 for transmission messages. In the latest version, 0.3 of the client sockets are bound to port 6666.

The reason that a client binds to a port is that he/she needs to have a standard port for both the relaying of traffic as well as the receiving of traffic this is because if the ports are different for these two operations, the server would be able to distinguish between the routes taken through the network by looking at and comparing port usage.

5.1.4. ROUTING

Routing plays an important part in the implementation. For the system to function, all traffic is routed into the tun0 interface. Using the following commands are for the client. See the following command :

```
$ sudo route add -net 0.0.0.0 netmask 0.0.0.0 metric 2 dev tun0
```

Furthermore, the current implementation struggles with ipv6 as the neighbour discovery protocol does not correctly work while proxying all traffic. Such that all ipv6 is disabled.
See the following command :

```
$ sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1
$ sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1
$ sudo sysctl -w net.ipv6.conf.lo.disable_ipv6=1
```

Furthermore, an individual route is added for all other peers, so the traffic isn't routed through the central server following command 1.

Lastly, To test this system, the following rules are added to allow SSH interaction with the client.
See the following command :

```
$ sudo ip rule add table 128 from $LOCAL_ADDR
$ sudo ip route add table 128 to $LOCAL_SUBNET/32 dev $DEFAULT_GATEWAY_NAME
$ sudo ip route add table 128 default via $DEFAULT_GATEWAY_IP
```


5.2. SYSTEM COMPONENT IMPLEMENTATION

This section shows the pseudo-code of the system components. Each system component is running multiple parallel processes. Each process has a distinct job, although these processes may share resources.

5.2.1. ID SERVER

The ID server does not handle packet requests. It monitors a client socket and a proxy (for the proxy server) socket for interaction with the respective system component. For each socket, a process is initiated to monitor said socket.

The system is implemented in such a way that clients are able to direct three different requests at the ID server. The requests Register ID, Register for groups are covered in section 4.4. The third request is Get IP which is required as otherwise the client does not know who to mention in the Client IP section of the packet.

The proxy handler process allows for three different types of requests from servers. First it validates the server is legitimate and then a server can register, add and update groups. This is a slight deviation from the algorithm as it is laid out in chapter 4 as this is easier to implement and the it should not influence performance.

5.2.2. PROXY SERVER

The proxy server, similarly to the ID server, monitors multiple file handlers using separate processes. In the case of the Proxy server, it monitors for incoming traffic from the peer net, it monitors outgoing traffic from the internet going to the peers and lastly, the socket for system interactions. The system relies on several backend databases, most importantly the session database. For each new combination of IP:port, a new session is

Algorithm 2 client handler process

```
1:  while True do
2:      packet_encrypted = client_socket.recvfrom(MAXIMUM_PACKET_SIZE)
3:      message = decrypt(packet_encrypted)
4:      ID_key = message['request']
5:      Timestamp = message['Timestamp']
6:      if client_legitimate(ID_key,Timestamp) then
7:          request = message['request']
8:          next_address = message['next_address']
9:          if request == 'Register_ID' then
10:             response = register_user()
11:         else if request == 'Get_IP' then
12:             response = get_user_ip()
13:         else if request == 'Register_for_groups' then
14:             response = user_group_registration()
15:         end if
16:         if response != None then
17:             response_encrypted = sym_encrypt(sym_key,response)
18:             client_socket.sendto(response_encrypted,next_address)
19:         end if
20:     end if
21: end while
```

generated by the client. The server keeps track of sessions such that internet responses can be sent back to the client.

Incoming data from the internet is matched to a session in line 3 and 4. If a session is found the exit information is extracted from the matched session and the response is symmetrically encrypted on line 10. The encrypted message is then sent to the specified next address.

An incoming message from the peernet is matched to a session and if no session is found a new session is generated this can be seen on line 11. Either way the request is decpsulated and placed on the TUN0 interface such that it can be handeld by the linux packet process.

Lastly the proxy server needs to handle requests for clients registering for groups 'group_registration' and clients who want to update their groups 'update_group'. for each of these requests the clients identity is verified using the group key as can be seen in line 7. Similarly to the other proxy requests the packet is encapsulated using the symmetric key on line 14 and then sent back to the requested address.

5.2.3. CLIENT

The client is the most complex piece of code as it acts as a proxy and a traffic initiator. Simmiarly to both other implemented systems a process is assigned to monitor each file descriptor. If a packet is requested by the client it is routed into the tun interface and enters the algorithm 'Client - outgoing request'. This algorithm aims to match the

Algorithm 3 proxy handler process

```
1: while True do
2:     packet_encrypted,address = server_socket.recvfrom(MAXIMUM_PACKET_SIZE)
3:     message = decrypt(packet_encrypted)
4:     ID_key = message['ID_key']
5:     Timestamp = message['Timestamp']
6:     server_pk = message['server_pk']
7:     if server_legitimate(ID_key,Timestamp) then
8:         request = message['request']
9:         if request == 'server_registration' then
10:            response = register_server()
11:        else if request == 'add_group' then
12:            response = add_group()
13:        else if request == 'update_group' then
14:            response = update_group()
15:        end if
16:        if response != None then
17:            response_encrypted = ECC25519(server_pk,response)
18:            server_socket.sendto(response_encrypted,address)
19:        end if
20:    end if
21: end while
```

Algorithm 4 Proxy server - internet traffic handler process

```
1: while True do                                  ▷ Internet -> Proxy Server -> Peer-net
2:     packet = tun0.read(MAXIMUM_PACKET_SIZE)
3:     session = match_packet_to_session(packet)
4:     if session then
5:         hash_list = session['hash_list']
6:         next_address = session['next_address']
7:         exit_info = session['exit_info']
8:         sym_key = session['sym_key']
9:         random_hash = hash_list.choose_random()
10:        encrypted_response = XSalsa20(sym_key,packet)
11:        final_packet = random_hash + encrypted_response + exit_info
12:        data_socket.sendto(final_packet,next_address)
13:    end if
14: end while
```

Algorithm 5 Proxy server - peernet traffic handler process

```
1:  while True do                                   ▷ Peer-net -> Proxy Server -> Internet
2:      packet_encrypted,address = data_socket.recvfrom(MAXIMUM_PACKET_SIZE)
3:      message = decrypt(packet_encrypted)
4:      ID_key = message['ID_key']
5:      sym_key = message['sym_key']
6:      Timestamp = message['Timestamp']
7:      request = message['request']
8:      if client_legitimate(ID_key,Timestamp) then
9:          session = match_packet_to_session(packet)
10:         if !session then
11:             session = generate_session(packet)
12:         end if
13:     end if
14:     write(tun0,request)
15: end while
```

Algorithm 6 Proxy server - system request handler process

```
1:  while True do
2:      packet_encrypted,address = system_socket.recvfrom(MAXIMUM_PACKET_SIZE)
3:      message = decrypt(packet_encrypted)
4:      ID_key = message['ID_key']
5:      sym_key = message['sym_key']
6:      Timestamp = message['Timestamp']
7:      if client_legitimate(ID_key,Timestamp) then
8:          request = message['request']
9:          if request == 'group_registration' then
10:             response = register_peer_for_groups()
11:         else if request == 'update_group' then
12:             response = fetch_update_groups()
13:         end if
14:         if response != None then
15:             encrypted_response = XSalsa20(sym_key,main_response)
16:             system_socket.sendto(response_encrypted,address)
17:         end if
18:     end if
19: end while
```

packet to a session and failing to do so generates a new one described in more detail in section 4.3. The packet is encapsulated according to the session and then sent to address specified by the session.

Algorithm 7 Client - outgoing request

```
1:  while True do                                  ▷ Client -> proxy server -> internet
2:      packet = os.read(self.tun_fd,MAXIMUM_PACKET_SIZE)
3:      session = match_packet_to_session(packet)
4:      if session == None then
5:          session = generate_session(packet)
6:      end if
7:      if session['mode'] == 'basic' then
8:          encrypted_packet = encapsulate_basic(packet,session)
9:      else if session['mode'] == 'bridge' then
10:         encrypted_packet = encapsulate_bridge(packet,session)
11:     else if session['mode'] == 'exit' then
12:         encrypted_packet = encapsulate_exit(packet,session)
13:     else if session['mode'] == 'full' then
14:         encrypted_packet = encapsulate_full(packet,session)
15:     end if
16:     if encrypted_packet != None then
17:         client_socket.sendto(encrypted_packet,address)
18:     end if
19: end while
```

If the client receives a message on its socket file descriptor that means a packet has arrived from the network. This packet could either be a proxy packet for another peer or it could be a return packet from the proxy server. To find which is which they are iteratively tried. In the case that it is a packet for another peer the packet is forwarded. otherwise the client matches the starting 10 characters to its hash list dictionary which indicates which symmetric key should be used to decrypt the message this can be seen on line 10 - 13.

Algorithm 8 Client - incoming request

```
1:  while True do
2:      packet_encrypted,address = client_socket.recvfrom(MAXIMUM_PACKET_SIZE)
3:      if bridge_packet(packet_encrypted) then   ▷ peer-net -> client -> Proxy Server ->
    Internet
4:          if verify_peer(packet_encrypted) then
5:              Proxy_packet(packet_encrypted)
6:          end if
7:      else if exit_packet(packet_encrypted) then ▷ Internet -> Proxy Server -> client ->
    Peer-net
8:          if verify_peer(packet_encrypted) then
9:              Proxy_packet(packet_encrypted)
10:         end if
11:     else                                          ▷ Internet -> Proxy Server -> client
12:         hash_id = packet_encrypted[:10]
13:         for session in sessions do
14:             if hash_id in session['hash_list] then
15:                 sym_key = session['sym_key']
16:                 break
17:             end if
18:         end for
19:         if sym_key != None then
20:             packet = XSalsa20(sym_key,packet_encrypted)
21:             pwrite(tun0,packet)
22:         end if
23:     end if
24: end while
```

5.3. TESTING SETUP

To test both latency and download speed of the system, Servers and simulated Clients are deployed to AWS in multiple countries. All servers and clients are deployed on EC2 instances with 1 CPU and 1 GB of Ram with low to moderate network performance. To simulate a variety of distances, we selected multiple countries that have a variety of distances from Europe (Germany); the countries chosen were Germany, Canada and China to simulate small, medium and large distances, respectively.

We estimate the latency of the deployed system using the ping command to find the Round Trip Delay (RTD) at various distances and using all different types of routes and the control case. The ping is sent to 8.8.8.8, which had a maximum response time of 1.5 ms from all clients. The mean is found of 100 ping latencies. This is not a perfect representation of the latency as the latency increase incurred using a different route would be due to distance primarily.

We estimate the system's download speed using the same construction as for the latency, but instead of measuring the RTD, the time is measured that a 100MB file is sent between a simulated client located outside of the system. The simulated client is located in the European cluster. The speed is not optimized to multiple threads, so it represents the baseline performance of the system.

6

Results & Evaluation

In this chapter, the Ethix system is evaluated and tested. First, Ethix is evaluated on its privacy and anti-censorship performance, with respect to the requirements laid out in chapter 3. Next, this chapter covers latency and download speed tests results, obtained with the implementation and the testing strategy outlined in section 5.3.

6.1. Privacy

in this section, we evaluate the privacy resilience properties of Ethix against malicious peers, proxy servers, control points, and GPAs. For each attacker, the relevant privacy attributes are evaluated along the lines of anonymity, unlinkability, undetectability and un-observability.

6.1.1. Malicious Peers

Peers cannot access unencrypted data, but that does not exempt them from producing potential privacy concerns. Peers proxy data for the client and are further aware of the client's existence. Against Ethix without fake packets, peers can determine which subject sent a message needing to be proxied and thus, they can break anonymity. Furthermore, multiple sessions can be linked to each other because peers know the sender of a message. Peers are unable to determine whether a transmission is legitimate traffic or fake packets, and thus they are unable to breach undetectability. Since both anonymity is broken, unobservability is not provided by the Ethix system against other peers. Since peers cannot tell whether a transmission is real or what a message contains, they do not pose a substantial risk to Ethix privacy.

If peers collaborate with other system components, such as the proxy server, this could lead to a potential fully de-anonymizing and unlinkability breaking attack.

6.1.2. Malicious Proxy server

Proxy servers are the only systems that can determine a message's contents are in plain text. This means that the privacy properties against proxy servers need to be particu-

larly strong. The anonymity property is maintained as the server cannot determine the sender of a message beyond the anonymity set (group). Furthermore, a proxy server cannot determine a link between multiple sessions (theoretically, a similar sender could probabilistically be determined using DPI and learning). The proxy server cannot determine whether a session is legitimate or fabricated, and thus undetectability is not broken. Lastly, the unobservability requirement is met as both anonymity and undetectability are maintained. If the server were to implement an information gathering system, the server might significantly generate an anonymity delta.

The differential privacy property of the group becomes apparent when the proxy server is considered an adversary. In the case discussed, the proxy server is honest but curious such that any data that is possible for the proxy server to link together will do so. The differential privacy property is such in that system that the proxy server cannot link a specific user (IP, ID key) as the sender or receiver of a packet with certainty. The packet arriving at the proxy server looks similar regardless of the route taken, and thus the proxy server cannot tell whether the sender or receiver is the legitimate sender or receiver. If the proxy server is honest but curious, the request it receives can be linked to a group key, symmetric key, entry IP, and exit IP. The group key can be mapped to a set of users and are thus not uniquely identifiable. Furthermore, the symmetric key is changed on a per session basis and thus cannot be mapped across sessions to identify a user. This leaves entry and exit IPs as possible identifiers of a specific user, and a unique identification is prevented as the proxy server will see one of two cases.

In the case that exit and entry IPs are the same in case the curious proxy server knows that the packets are either sent in basic route or full route. This knowledge either means that the sender and receiver IP are either the user in the case of the basic route or are not the user in the case of the full route. This means that the proxy server cannot definitively say who the user is who sent the request.

If the exit and entry IPs are different, the curious proxy server knows that the packet was sent either in the bridge or exit route. Since, in this case, the proxy server does not know whether the user is the exit or entry IP, it cannot definitively say who the user is who sent the request.

Lastly, fake packets would allow completely fictitious packets to be injected into the network as described in the packet format section. This would result in neither the sender nor receiver IP being valid, and thus the proxy server could not be sure even that the traffic is coming out of a group.

Interestingly, since the clients choose the routes percentages based on their needs, the percentages that each group has will vary, and since the proxy server is unable to determine which route is being used, even groups where 100 % of the interactions are using a basic route the proxy server would be confused. This means that this system could theoretically achieve latency as low as classical VPNs while providing higher privacy.

6.1.3. CONTROL POINTS

Control points are generally located between the peer and the proxy server. These systems would be unable to determine the originator of a message beyond the anonymity set. Furthermore, control points would be unable to breach unlinkability as session packets look entirely different. Like all other systems, undetectability would be difficult

to breach as it would be difficult to determine dummy packets from real ones. This results in unobservability from the perspective of control points as the other requirements are met.

6.1.4. GPA

The global passive adversary is located everywhere on the internet. They can monitor all connections. Even with using false, mixing and splitting modifications observations can be made to reduce to total anonymity of the user. GPA could increase the correctness of their guessing by using observations but would be unable to determine who the sender of a particular message is. Thus, anonymity or at least partial anonymity is provided. The GPA would have to monitor the whole group independently to be sure of its ability to link multiple sessions together. Even the GPA cannot determine whether a packet is legitimate or a dummy message since both anonymity and undetectability hold. The GPA is unable to breach unobservability. There are probabilistic indicators that could allow the GPA to become confident in guessing the originator of the message.

6.2. CENSORSHIP EVASION

The censorship evasion of this system is evaluated by comparing it with the requirement analysis. The evaluation is made against the currently most sophisticated firewall, the GFW. In summary, Ethix currently is effectively able to deal with all current censorship methods. This is subject to change in deployment as Tor and VPN-gate has faced additional issues resulting from the GFW attacking their system specifically. These attacks are difficult to predict, and thus Ethix cannot be evaluated against them.

6.2.1. PLAIN TEXT TRAFFIC ATTACKS

Since traffic is always sent over proxies and encrypted plain text, traffic attacks should be impossible against the Ethix system. This matches the requirements laid out for the system in chapter 3.

6.2.2. IP BLOCKING

IP blocking is a highly effective way to block VPNs, Tor and other services. IP blocking is generally an automated procedure. These automated procedures need to have a low margin of error as if IPs are incorrectly blocked, damage is incurred in terms of productivity or system failure. Therefore, these systems need high accuracy data. This data can be found from publicly available sources and private sources. If there are few ways for a censor to gain IP information IP blocking attacks fail. The following sections discribe how Ethix prevents public and private IP enumeration.

PUBLIC IP ENUMERATION

Public IP enumeration is effective against systems that have most of their proxy services available online. Ethix hides the vast majority of its services from the public. The only proxy services visible online would be a small subset of nodes acting as system entry points. These points, similarly to Tor, can be hidden behind CAPTCHAs and linked to an IP location such that servers behind a subnet would be unable to enumerate public

nodes efficiently.

PRIVATE IP ENUMERATION

Private IP enumeration could be most efficiently done in Ethix by signing up to the network and recording peers in shared groups. The key server can mitigate this attack by preventing excessive group switching and could further be reduced by making group sizes smaller (this would come at the cost of privacy).

6.2.3. QOS FILTERING

QoS filtering is the newest and most advanced component of the GFW as it can learn and classify protocols. Generally, two approaches can be used to fingerprint data that censors want to block.

The first approach is carried by fingerprinting the connection, which can be done by monitoring the ratio of data exchanged between parties, ports used, and packets sent in specific frequencies. In Ethix, all of these parameters can be tweaked: a range of different ports could be used, data could be sent to peers and servers with different distributions, or the system sign up could be altered.

The second approach uses packet fingerprinting. Tor was initially targeted by packet fingerprinting this by the GFW to mitigate this Tor developed pluggable transports [10], [11]. Pluggable transports disguise traffic between 2 users to look like non-suspicous traffic traffic. The current state of the art is obfs4, and it could be deployed between clients and proxy peers. Ethix in its current state is not resilient to this, but if we implemented obfs4 between clients and proxies, we would be resilient.

6.3. LATENCY

Sufficiently low latency is a requirement of any use able private system as many applications rely on fast feedback loops. Thus Ethix is tested as described in section 5.3. Looking at fig. 6.1, the first set of bars show the latentencies when routing traffic with the Frankfurt AWS data center. The overhead latency of the system within the same data center sat around 3.5 - 5 ms, depending on what type of route is used. This variation of 1.5 ms shows that there is little latency overhead when more proxy stops are added like in the full route. Latency increased significantly when a client based in canada see second set of bars proxies traffic through frankfurt (germany). The second set of bars shows some variation between the route latencies this could be due to the natural variation of the network. Lastly the third set of bars shows a noticeble increase in latency and little variation between the routes. Interestingly, in the case where the server and client have located far apart, the variation of distance eclipses the variation due to the additional latency of the full route.

In fig. 6.1 the latency's by distance show that the most significant factor in the system's latency is the distance travelled. The results are aligned with global ping statistics [25]. The case where the servers are close together can evaluate the additional latency added through each jump. The overhead latency of the system with no distance (same data centre) at around 3.5 - 5 ms, depending on what type of route is used. This shows that the proxy the system do system implementation do not have excessive overhead appart from the additional distance traveled.

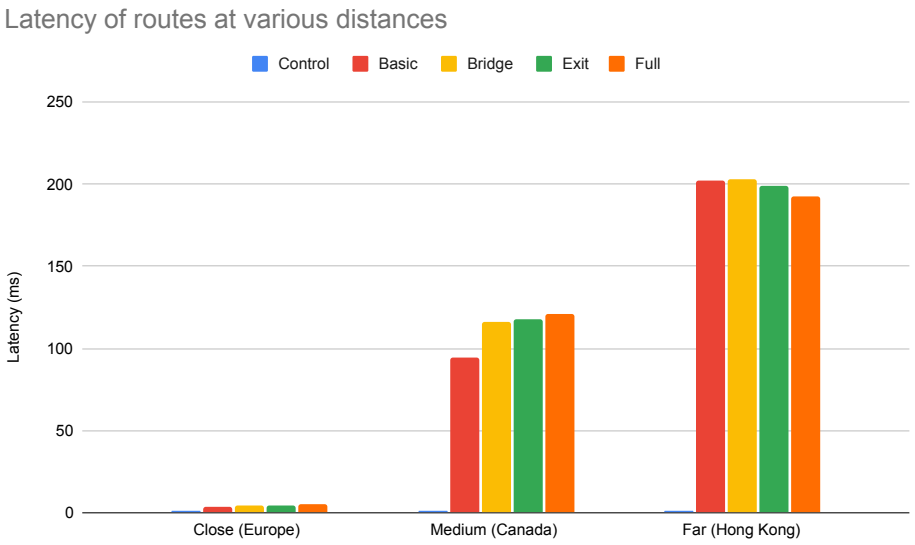

Figure 6.1: Latency tests preformed at three different distances with all data packet routes.

As is shown by the fig. 6.1 latency scales with distance travels. While this system was not tested in a live setting, this implies that in general, the number of times proxying occurs and the further the proxy are separated from each other, the larger the latency grows. Ethix is designed to take minimal required jumps while maintaining the session mixing property and differential privacy. This, in theory, should result in lower latency than other systems. Furthermore, the distance between proxy's (peers and servers) decreases with an increase of users, and thus the system should get faster with the number of users in the network.

6.4. DOWNLOAD SPEED

Similarly to latency, download speed is a requirement for any modern privacy/censorship evasion system. To verify that Ethix has sufficient download speed the Ethix system is tested according to the testing setup described in section 5.3. The tested results are shown in fig. 6.2. The first set of bars shows that routing through the Ethix network at a close distance results in a reduction by a factor of six versus the control case. Furthermore, there is little variation between the download speed through Ethix depending on the route taken with all four routes showing download speeds of around 20 - 22 Mbps. The second set of bars shows the download speed from Europe to Canada and a noticeable decrease in the control case speed can be noted vs the Europe to Europe download. The download speeds through the Ethix network is simmilar to the download speeds at close distance ranging from 14 - 18 Mbps. Lastly, the third set of bars show the download from Europe to Hong Kong. The control download test further decreases with distance and there is more variation in the download speeds depending on the route taken. In-

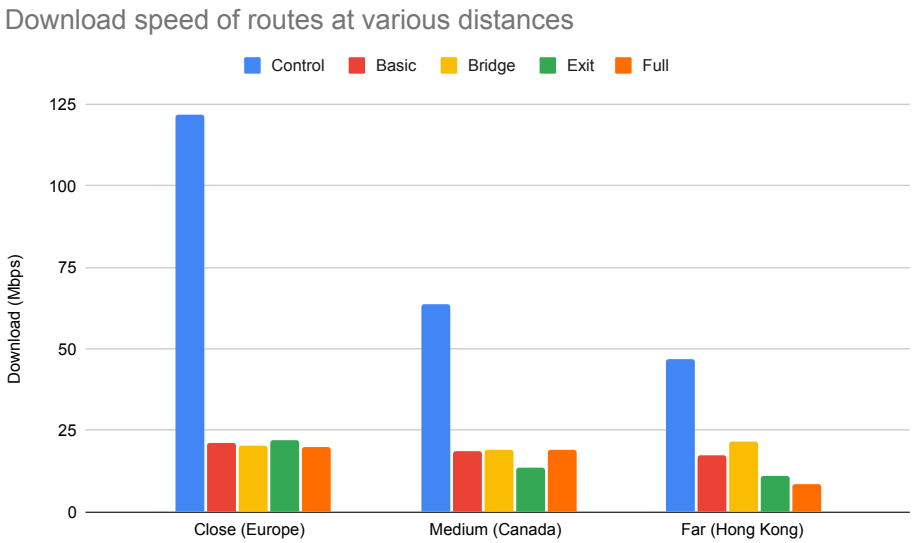

Figure 6.2: Throughput tests preformed at three different distances with all data packet routes.

terstinly both the exit and the full route are significantly slower. This could be due to the download file moving through extra proxies. Although this trend isent shown in either Europe to Europe or Canada to Europe download speeds.

While the overall speeds where lower then expected an average 16mbps download speed sufficient for regular internet navigation and even watching videos online. The transfer speed that that came out of the tests are slower than expected; we suppose it is an I/O bandwidth limitation on the AWS EC2 instances because multithreading and multiprocessing do little to increase the speed, and the way to decrease the speed is to use it print statements. The theory of an I/O bottleneck is furthermore encouraged by the roughly consistent transfer speeds at 16 Mbps which if the system was not bottle necked we would expect to change with the number of proxies in a route as well as with distance. Furthermore, as predicted in chapter 2 download speed clearly scales with distance as can be seen by the three times larger transfer rate in the control in Europe vs the control in Hong Kong.

Theoretically, the system should have a higher data transfer rate than VPN Gate, Nym network and Sentinel as Ethix aims is to minimize the number of jumps through the system and primarily through peer to peer systems. Furthermore, an increase in download speed is expected if more clients are added to the system as this would allow the multiplexing of data streams through multiple clients for individual session.

7

RELATED WORK & COMPARISON

The system proposed in this paper aims to provide two things privacy and censorship evasion. The related work aims to achieve some variation of those goals and generally fall into four categories Centralized VPNs, Decentralized VPNs, Hybrid Systems and Anonymity systems. Please note that this taxonomy may put some research into more than one category. Furthermore, we evaluate the specific systems on the properties of table 7.1. These evaluations abstract away information and serve as a general estimation of a system rather than a scientific evaluation. The metrics that are used include the number range 0 - 10 for privacy and censorship resistance. A system is evaluated for the worst-case attacker and for each component a score is assigned on the number range 0 - 2. 0 for the system not having the property and 2 being the system having the property.

Privacy evaluation components

1. Anonymity
2. Unlinkability
3. Undetectability
4. Unobservability
5. No-Few Privacy attacks available

Censorship evasive evaluation components

1. Clear traffic attacks not-available
2. Public IPs hidden
3. Privat IPs hidden
4. Active probing resistant
5. QoS filtering resistant

Table 7.1: Comparison table of related work. *General use is evaluated by the criteria whether all traffic (ICMP, UDP, TCP) to anyone can be sent through the system

	<150ms Latency	Low Overhead	General use*	Client Privacy (0-10)	Censorship resistant (0-10)
Ethix	✓	✓	✓	6	9
Nord VPN	✓	✓	✓	2	4
TOR	X	✓	X	6	10
VPN Gate	✓	✓	✓	6	6
Orchid VPN	✓	✓	✓	4	4
NYM	X	X	✓	10	10
VPN zero	✓	✓	X	10	N/A
I2P	X	X	X	2	2
Loopix	X	X	X	10	N/A

7.0.1. CENTRALIZED VPNS

VPNs generate an encrypted tunnel between a centralized server and the client's device. The client forwards traffic into the tunnel, and the server forwards the traffic and once a reply is received, forward it back through the tunnel. This architecture introduces little overhead as the centralized server is generally placed at an internet exchange point and uses high-quality hardware. In terms of the trust model, VPN servers can inspect all traffic over them. This results in high granularity insights into client data and activities. This means that clients have to trust that VPNs will not abuse their data. To affirm that VPN providers are trustworthy many VPN providers claim to keep no logs. That VPN providers stick to this promise is difficult to verify and, when verified, have shown that sometimes logs are kept despite the claim [26].

NORD VPN

Nord VPN is one of the largest commercial VPN providers. They use state of the art centralized VPN technologies and will act as a prototype of all VPNs. When **evaluated** the system adds little latency, overhead and is generally useable. Nord VPNs privacy potential worst attacker is the VPN server. Although Nord VPN has been the third party verified not to record data and store it in RAM, the VPN server is still the worst-case attacker. The VPN server can break all privacy evaluation components but has a small attack surface, so few privacy reducing attacks are available. The Censorship evasive properties of VPNs are that no apparent traffic attacks are available as the data is encrypted. Public IPs are scrapable. Private IPs are non-existent and Active probing can be performed on the system. Lastly, an emerging trend in VPNs is pluggable transports, and Nord VPN uses some traffic obfuscator and thus is QoS resistant.

7.0.2. Hybrid VPNs

Hybrid VPNs are made up of a combination of centralized controlled components and decentralized peer to peer components. These systems benefit from having access to a more significant number of IPs (P2P components) and more trusted system management components like the Tor directory nodes. The hybrid VPNs that are examined further in this section are Tor and VPN gate.

Tor

Tor is currently the largest and most widely used anonymous network. It has six thousand nodes and an estimated eight million daily users [22]. The six thousand nodes are volunteer-run, but the network has a centralized backbone as it uses ten hardcoded directory servers. The Tor protocol was designed in 2002 and based on the onion routing protocol [27]. The client fetches a list of active nodes and their keys. The client then chooses typically three nodes and builds a multi-hop connection. The keys are then used to layer encryption so that each node in the multi-hop can remove one layer of encryption. This decryption is forwarded to the next node until the last node, which can read the request and execute it. This construction results in only the first node knowing the client's identity and only the last node knowing the destination. While it protects the user's privacy much better than VPNs, global adversaries who can monitor both the exit and the entry node can perform an end to end correlation attack [28]–[30]. Tor was developed in 2002 and, as one of the oldest systems, has been vetted and analyzed, resulting in the finding of more than 50 de-deanonymization attacks against Tor [6]. **Evaluating** Tor using the Tor metrics page [5] shows that latencies mean range from 70 - 600 ms depending on the location. With throughput reaching 42 Mbps but averaging less than 10 Mbps. Tor itself is not a general-use application as it only provides browser functionality. The Strongest attacker for Tor would be the exit server without using attacks unable to break anonymity. Furthermore, it cannot link multiple user sessions together. There is no fake packet generation and thus undecidability is not present. Tor has been fighting the censorship war for the last two decades and is one of the most sophisticated applications against censorship, increasingly so with the recent release of snowflake [17]. Which is a large collection of volunteers that offer an entry into the Tor network.

VPN Gate

VPN Gate is a public VPN relay service designed to be censorship-resistant, specifically against the Greate Firewall (GFW) [13]. VPN gate organizes volunteers who provide a service of proxying traffic to evade censorship. This would be simple to block with IP blackholing if it were not for the novel ways to deter this. The first way to deter blocking is the addition of innocent IPs to the public directory, and the second is to prevent a network enumeration attack by using a central list of spies that records and blocklists IPs that request too many nodes. It was launched in 2013 and has facilitated 464 000 connections. 70 % of the volunteers remained unblocked at the time [13] of the paper being written. The system is still available today, but little is known about the percentage of unblocked volunteers. **Evaluating** the VPN Gate 2013 paper shows that it has low latency low overhead and can be used generally [13]. There are currently few publicly known attacks against VPN gates privacy. The censorship resistance of the VPN gate provides resistance against non-encrypted traffic attacks. Furthermore, it provides client

anonymity and session unlinkability. Similarly to Tor, the last two privacy categories that do not apply (undetectability, unobservability) are rated as neutral as there is no fake packet injection. Furthermore, although all IPs are public, they are protected using the spy list, which prevents active probing. Lastly, the VPN gate does not provide pluggable transport and thus is vulnerable to QOS attacks.

7.0.3. FULLY DECENTRALIZED VPNS

This section covers fully decentralized vpns and examines further Orchid, NYM, and VPN zero. Recently Orchid and Sentinel have tried to address the problem of centralized trust by creating the first decentralized VPNs (dVPNs). These systems incentives bandwidth donors with crypto-currency, which clients can buy to send traffic through the system. The current state of the art is proof that the concept of a dVPN works but are susceptible to Traffic analysis and censorship [31]. With Orchid stating as a limitation, "The Orchid node directory is published on the Ethereum blockchain and thus is public to the world." [32]. These designs have been improved upon by later designs such as the Nym network.

All crypto dVPNs mentioned above share the property that there is an equally privileged node that releases the traffic into the internet. This has potential problems as they could be acting as a proxy for illegal data (Child pornography, Dark markets). Furthermore, exit nodes can manipulate traffic in attacks similar to Tor exit node attacks, resulting in the de-anonymization of the client. A new dVPN by the creators of Brave has aimed to fix these problems (illegal traffic proxying and exit node attacks.) called VPN zero.

ORCHID

Orchid is a decentralized market for VPNs. It works using a blockchain that releases nano payments for traffic proxied. For the security of the proxying, it uses multi-hop onion routing similarly to Tor. This results in it importing a couple of vulnerabilities associated with Tor as well. **Evaluating** Orchid has to be done via third party accounts or research. 3rd party accounts mention 30 ms Latency, decent throughput, and the implementation seems to be generally useable. The privacy properties that Orchid provides are impressive for a VPN providing anonymity and unlinkability from the exit proxy. Undetectability and unobservability are provided on the level of payments as they are probabilistic, but there are known attacks against onion routing. Sadly, the censorship evasion properties are a little more sparse as all IPs are public and easily scrapeable. Non-encrypted traffic attacks are not viable, and obfuscating traffic could be implemented but is not currently.

NYM NETWORK

Nym network is a decentralized authentication and payment protocol founded on Mixnet, which iterates upon Tor. Users send all data through a mixed network using in the form of a sphinx packet [33] which keeps the data uniform. Furthermore, the mixnet adds fake messages and adds delays similar to Chaum mixes [34]. The traffic going into mixnet is therefore highly obfuscated and impossible to track back, even for GPAs. Evaluating the Nym network is currently not fully possible as the white paper does not give sufficient information. NYM network is based on Loopix, which had a 1.9 second mean RTD

time and further, it mentions using a diverse network of 10 mixes which generates large amounts of overhead. Nym network is a general use, and the privacy and theoretical censorship-resistant properties are currently state-of-the-art.

VPN ZERO

VPN zero aims to fix the exit node attack by using TLS to certify users visit a specific domain. It further solves logging by providing never moving unencrypted data over the system. It is fully decentralized and offers privacy and security guarantees which previous dVPNs have not. Since it relies on TLS, it allows only access to computes with an x509 certificate, and this limits VPN zero in its general use. [35]. **Evaluating** VPN zero indicates that the latency and overhead increases of using the system are negligible. The system only allows access to specific websites on its allowlist. It provides the overall highest privacy of any system as clear data is never accessible for a system component, but the paper [35] does not indicate how its network is un-sufficient detail to determine censorship resistance.

I2P

The "Invisible Internet Project" (I2P) is a fully decentralized alternative to Tor. In a similar manner to Tor clients, I2P clients build multi-hop connections where data is encrypted in layers once per hop [36]. Differently from Tor, I2P replaces the centralized directories of Tor with a distributed hash table (DHT) as well as having shorter unidirectional tunnels instead of the long-lived bi-directional tunnels of Tor. I2P is open to a variety of attacks that could allow an attacker to censor, disconnect, misdirect and deanonymize users [37]–[41]. **Evaluating** IP2 shows that high latency and low throughput are issues that the system has. Furthermore I2P is not a generally usable system and due to attacks mentioned in [37]–[41] I2P is not private or censorship-resistant.

7.0.4. PRIVACY NETWORKS

Privacy networks are networks designed to provide some service to the user with a high level of privacy. There have been a variety of privacy networks developed, with early versions being Chaum mixes [34], and later versions adding generality and functionality such as Loopix [42], Vuvuzela [43], and Herd [44].

LOOPIX

Loopix is an anonymous communication system that achieves high anonymity levels by using cover traffic and timing mixes. The Loopix system is made up of Service providers and mix nodes. Service providers are acting as entry gateways into the Loopix network [42]. It has low latency for a system using mix-networks with a mean of 1.93 seconds latency. **Evaluating** Loopix indicates that, similarly to Nym, it has high latency and high overhead, but it is the non-general implementation of Nym.

7.1. COMPARISON

Comparing Ethix to the systems discussed in the related work requires evaluating it. Ethix is evaluated according to the metrics set out for table 7.1. Ethix, according to the testing performed, has low latency increases and low overhead. Furthermore, since Ethix

is built on top of the IP layer, it is generally useable for most traffic. It provides anonymity from the exit proxy such that the exit proxy is unable to track sessions to individual users. Furthermore, unlinkability is provided between sessions as long as sessions are not linkable to each other. The use of fake packets increases undetectability, and thus unobservability is logically there as well. While no known privacy attacks are available since this system is not deployed, and the privacy properties are not perfect. Thus the client privacy of Ethix is graded at a six. The censorship resistance of Ethix is graded at a nine as no traffic attacks are available, public IPs are hidden behind either, or CAPTCHA is handed out selectively in the case of proxy servers. Next, it is difficult to enumerate private IPs since the ID server has several mechanisms to prevent this. Furthermore, the system is active probing resistant as clients and proxy servers do not reply to non validated systems. Lastly, while the system could be QoS filtering resistant using pluggable transports and protocol warping, this is not the case in the current implementation, and thus the system partially meets the criteria for QoS filtering resistance.

As can be seen on table 7.1 Tor, Nym, and Loopix fail to serve the general customer as more significant latency, and throughput overheads are present. Furthermore, Tor, VPN zero and Loopix do not allow all IP traffic to be used by the system. These properties prevent a variety of use cases for normal consumers. These systems specialize in being censorship-resistant in the case of Tor and Nym or privacy guaranteeing in the case of VPN zero, Loopix and Nym.

Nord VPN, Orchid VPN provide low latency and low overhead while being generally useable for all IP traffic, making them consumer-friendly. This comes at the cost of client privacy and censorship resistance, making these systems difficult to use in countries where VPNs are blocked.

Two outliers try to provide consumer useability as well as privacy and censorship resistance. These outliers are Ethix and VPN gate. VPN Gate is already active and run by volunteers. Since it is an academic project, it cannot advertise and scale properly as no funds are available. In contrast, Ethix could be deployed in a for-profit setting. Furthermore, since all IPs of servers are publicly available for VPN gate, censorship resistance is difficult to provide if some way is found to efficiently determine the fake IPs from the legitimate ones as a point of comparison; Ethix aims to prevent all IPs from being leaked. Lastly, VPN gate uses per-packet four hops (client-> proxy -> VPN server -> proxy -> client) while Ethix depending on the configuration, uses per-packet an average of two-four hops per packet. This could result in Ethix performing better and using the volunteers more efficiently, and in the proxy server gaining more information to break anonymity with.

8

CONCLUSION

The system proposed in this thesis is not fully developed and designed. This section talks about possible areas that could be expanded on and the limitations of the research.

8.0.1. FUTURE WORK

1. Packet format: Increase to multiple group proxy hops
2. Peer incentives: Design a theoretical peer incentives system.
3. Deployment: Deploy system and test system in a less controlled setting
4. Deployment: Fully multithreading the system
5. Deployment: Hide proxy server IPs
6. Theory: network overhead depending on client route configuration including splitting and false packets
7. Theory: Average hop length depending on client route configuration
8. Theory: Anonymity delta calculations depending on client route configuration

8.0.2. LIMITATIONS

There are several limitations within this study: the system is not tested in a real-world scenario. This is a major limitation as government censoring is generally targeted at specific systems and thus tailored to them [6], [28]–[30], [37], [39]. While the system proposed in this paper has theoretically good abilities against censoring, this is not tested in practice, and thus this claim can not be verified fully.

A further limitation of this work is that plain text traffic going over the proxy server could be fingerprinted to link multiple sessions together. This could be achieved for instance using the browser header. While these packets could be encrypted using TLS there is a possibility for fingerprinting even this.

The use of fake packets is said to increase undetectability. It could be possible to determine weither a packet is fake or not if the fake packet generator was not sophisticated. Making a sophisticated fake packet generator is not part of the scope of this research and while papers exist that do this their success rate is at around 79% for HTTP packets and these are simple HTTP get requests [45]. This could limit the viability of the fake packet generator for Ethix.

The limitation of not being a real-world deployment further limits the predictive capabilities from the results for download speed and latency. The testing was done on similar weak hardware that may not accurately model a real-life scenario. This means that there may be a significant deviation in latency and download speed from the numbers in this paper if strong hardware and real clients are used.

The last limitation mentioned in this paper comes again from the system described in this paper is a novel one, and as such, there will be vulnerabilities unforeseen. These vulnerabilities could compromise the core functionality of the system and could theoretically invalidate the entire concept.

8.1. CONCLUSION

This research aimed to find a system that was usable by everyday consumers who prefer a lower latency, censorship-resistant solution. The Ethix system is, in theory, a highly efficient, censorship-resistant and encouraging privacy system. It could be used to replace private VPN usage, but since this system was not tested in a live setting, this is purely in theory. If it can replace VPNs, this could result in a globally censorship-resistant community with very low latency.

BIBLIOGRAPHY

[1] S. Rose, *Russia bans nord vpn and express vpn*, https://www.bloomberg.com/news/articles/2021-09-03/russia-blocks-nord-vpn-express-vpn-in-bid-to-control-content, Bloomberg, 2021.

[2] S. Pai, *Russia-bans-six-vpn-providers*, https://developingtelecoms.com/telecom-business/telecom-regulation/11840-russia-bans-six-vpn-providers.html, Developing Telecoms, 2021.

[3] S. Schaub, *The blocking of vpns in india and china spells trouble for our internet freedom*, https://www.techradar.com/news/the-blocking-of-vpns-in-india-and-china-spells-trouble-for-our-internet-freedom, Techradar, 2019.

[4] S. Dogra, *What is vpn, and why does home ministry want to ban it? here is all you need to know*, https://www.indiatoday.in/technology/features/story/what-is-vpn-and-why-does-home-ministry-want-to-ban-it-here-is-all-you-need-to-know-1848397-2021-09-02, India Today, 2021.

[5] *Tor metrics*, https://metrics.torproject.org/onionperf-latencies.html, Cisco press, 2001.

[6] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam, and S. Jha, "Anonymity with tor: A survey on tor attacks," *arXiv preprint arXiv:2009.13018*, 2020.

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

[8] constituteproject, *Iran (islamic republic of)'s constitution of 1979 with amendments through 1989*, https://www.constituteproject.org/constitution/Iran_1989.pdf?lang=en, constituteproject, 2021.

[9] M. T. Khan, J. DeBlasio, G. M. Voelker, A. C. Snoeren, C. Kanich, and N. Vallina-Rodriguez, "An empirical analysis of the commercial vpn ecosystem," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 443–456.

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

[11] ssteele, *Tor at the heart: Bridges and pluggable transports*, https://blog.torproject.org/tor-heart-bridges-and-pluggable-transports, Tor, 2021.

[12] F. Douglas, W. P. Rorshach, W. Pan, and M. Caesar, "Salmon: Robust proxy distribution for censorship circumvention.," *Proc. Priv. Enhancing Technol.*, vol. 2016, no. 4, pp. 4–20, 2016.

[13] D. Nobori and Y. Shinjo, "{vpn} gate: A volunteer-organized public {vpn} relay system with blocking resistance for bypassing government censorship firewalls," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 229–241.

[14] C. Arthur, *China tightens 'great firewall' internet control with new technology*, https://www.theguardian.com/technology/2012/dec/14/china-tightens-great-firewall-internet-control, The Guardian, 2013.

[15] A. Troianovski, *Google and apple, under pressure from russia, remove voting app*, https://www.nytimes.com/2021/09/17/world/europe/russia-navalny-app-election.html, The Guardian, 2021.

[16] *What's the best vpn for china? we tested 59 to see which work*, https://www.comparitech.com/blog/vpn-privacy/whats-the-best-vpn-for-china-5-that-still-work-in-2016/, Paul Bischoff, 2001.

[17] K. MacMillan, J. Holland, and P. Mittal, "Evaluating snowflake as an indistinguishable censorship circumvention tool," *CoRR*, vol. abs/2008.03254, 2020. arXiv: 2008.03254. [Online]. Available: https://arxiv.org/abs/2008.03254.

[18] E. A. Pfitzmann, *Terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management draft-hansen-privacy-terminology-00.txt*, https://tools.ietf.org/id/draft-hansen-privacy-terminology-00.html, ietf, 2011.

[19] *Implementing quality of service over cisco mpls vpns*, https://www.ciscopress.com/articles/article.asp?p=471096&seqNum=6, Cisco press, 2001.

[20] M. Ehlert, "I2p usability vs. tor usability a bandwidth and latency comparison," in *Seminar Report, Humboldt University of Berlin*, 2011, pp. 129–134.

[21] A. Barberis, D. Radosevich, W. Emery, and M. Borowczak, "Portable tor router: Easily enabling web privacy for consumers," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2018, pp. 1–6.

[22] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr, "Understanding tor usage with privacy-preserving measurement," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 175–187.

[23] L. Lu, J. Han, Y. Liu, L. Hu, J.-P. Huai, L. Ni, and J. Ma, "Pseudo trust: Zero-knowledge authentication in anonymous p2ps," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 10, pp. 1325–1337, 2008.

[24] O. Titz, *Why tcp over tcp is a bad idea*, http://sites.inka.de/~W1011/devel/tcp-tcp.html, Olaf Titz, 2001.

[25] WonderNetworks, *Global ping statistics*, https://wondernetwork.com/pings, WonderNetworks, 2021.

[26] A. Martin, *Lulzsec hacker exposed by service he thought would hide him*, https://www.theatlantic.com/technology/archive/2011/09/lulzsec-hacker-exposed-service-he-thought-would-hide-him/337545/, 2011.

[27] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *International workshop on information hiding*, Springer, 1996, pp. 137–150.

[28] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 337–348.

[29] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *International Conference on Financial Cryptography*, Springer, 2004, pp. 251–265.

[30] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell-counting-based attack against tor," *IEEE/ACM Transactions On Networking*, vol. 20, no. 4, pp. 1245–1261, 2012.

[31] M. Backes, G. Doychev, and B. Köpf, "Whitepaper - a blockchain framework for building decentralized vpn applications," in *NDSS*, SNT Foundation, 2013.

[32] J. S. Cannell, J. Sheek, J. Freeman, G. Hazel, J. Rodriguez-Mueller, E. Hou, B. J. Fox, and S. Waterhouse, "Orchid: A decentralized network routing market," Orchid Labs, Tech. Rep., 2019.[Online]. Available: https://www. orchid. com . . ., Tech. Rep., 2019.

[33] G. Danezis and I. Goldberg, "Sphinx: A compact and provably secure mix format," in *2009 30th IEEE Symposium on Security and Privacy*, IEEE, 2009, pp. 269–282.

[34] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[35] M. Varvello, I. Q. Azurmendi, A. Nappa, P. Papadopoulos, G. Pestana, and B. Livshits, "Vpn-zero: A privacy-preserving decentralized virtual private network,"

[36] B. Zantout, R. Haraty, *et al.*, "I2p data communication system," in *Proceedings of ICN*, Citeseer, 2011, pp. 401–409.

[37] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, "Practical attacks against the i2p network," in *International workshop on recent advances in intrusion detection*, Springer, 2013, pp. 432–451.

[38] S. H. Jeong, A. R. Kang, J. Kim, H. K. Kim, and A. Mohaisen, "A longitudinal analysis of. i2p leakage in the public dns infrastructure," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 557–558.

[39] N. P. Hoang, P. Kintis, M. Antonakakis, and M. Polychronakis, "An empirical study of the i2p anonymity network and its censorship resistance," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 379–392.

[40] M. Herrmann and C. Grothoff, "Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p," in *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, 2011, pp. 155–174.

[41] E. Erdin, C. Zachor, and M. H. Gunes, "How to find hidden users: A survey of attacks on anonymity networks," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2296–2316, 2015.

[42] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1199–1216.

[43] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015, pp. 137–152.

[44] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merritt, "Herd: A scalable, traffic analysis resistant anonymity network for voip systems," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 639–652.

[45] A. Cheng, "Pac-gan: Packet generation of network traffic using generative adversarial networks," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019, pp. 0728–0734. DOI: 10.1109/IEMCON.2019.8936224.

A

APPENDIX

A.1. ENCAPSULATION ALGORITHMS

The following algorithms are the algorithms to encapsulate data in the basic, bridge, and exit format.

Algorithm 9 Basic Route Encapsulation

Input: message,server_pubkey,sym_key
Output: encrypted_packet

```
main_message,exit_message = generate_content(message)

                                                    ▷ Encrypt message
encrypted_sym_key = ECC25519(server_pubkey,sym_key)
encrypted_message = XSalsa20(sym_key,main_message)
encrypted_main = encrypted_sym_key + encrypted_message

                                                ▷ Encrypt fake exit message
encrypted_fake_exit_message = ECC25519(dummy_pubkey,random_string)

                                                ▷ combine components and return
encrypted_packet = encrypted_main + encrypted_fake_exit_message
```

A

Algorithm 10 Bridge Route Encapsulation

Input: message,server_pubkey,bridge_pubkey,sym_key
Output: encrypted_packet

```
bridge_message,main_message,exit_message = generate_content(message)

                                                    ▷ Encrypt bridge message
encrypted_bridge_message = ECC25519(bridge_pubkey,bridge_message)

                                                    ▷ Encrypt main message
encrypted_sym_key = ECC25519(server_pubkey,sym_key)
encrypted_message = XSalsa20(sym_key,main_message)
encrypted_main = encrypted_sym_key + encrypted_message

                                                    ▷ Encrypt fake exit message
encrypted_fake_exit_message = ECC25519(dummy_pubkey,random_string)

                                                    ▷ combine components and return
encrypted_packet = encrypted_bridge_message + encrypted_main + en-
crypted_fake_exit_message
```

Algorithm 11 Exit Route Encapsulation

Input: message,server_pubkey,exit_pubkey,sym_key
Output: encrypted_packet

```
main_message,exit_message = generate_content(message)

                                                    ▷ Encrypt message
encrypted_sym_key = ECC25519(server_pubkey,sym_key)
encrypted_message = XSalsa20(sym_key,main_message)
encrypted_main = encrypted_sym_key + encrypted_message

                                                    ▷ Encrypt exit message
encrypted_exit_message = ECC25519(exit_pubkey,exit_message)

                                                    ▷ combine components and return
encrypted_packet = encrypted_main + encrypted_exit_message
```