Hiding Content in Video Calls to Resist Censorship

Raúl Aguilar Arroyo, 429631

Abstract

Internet censorship restricts the fundamental right to freedom of expression and access to information, particularly in regions governed by authoritarian regimes. This paper explores the potential of leveraging WebRTC-based video calls to circumvent censorship by analyzing six notable protocols: SkypeMorph, CensorSpoofer, Stegozoa, Protozoa, Snowflake, and TorKameleon. These protocols use techniques such as steganography, protocol mimicry, and dynamic proxy networks to hide or disguise censored content within video calls. The survey evaluates the efficacy, performance, and vulnerabilities of each approach, highlighting their strengths and limitations in real-world applications. This analysis provides insights for developing more resilient censorship circumvention tools and offers guidance to users seeking secure methods for bypassing censorship.

1 Introduction

Censorship of online content is an escalating issue worldwide, threatening the principles of free expression and unrestricted access to information. Authoritarian governments employ sophisticated methods—such as deep packet inspection, URL filtering, and protocol disruption—to control internet activity and suppress dissent. These measures have severe implications for access to critical information, communication, and knowledge-sharing platforms, impacting millions of users.

In response, researchers are investigating new methods to resist censorship effectively. WebRTC-based video calls provide a promising avenue due to their encrypted, peer-to-peer nature and widespread adoption in legitimate applications. By embedding or disguising censored content within video streams, these methods make detection and blocking by censors significantly more challenging. This paper surveys six prominent protocols—SkypeMorph, CensorSpoofer, Stegozoa, Protozoa, Snowflake, and TorKameleon—that leverage video calls to achieve censorship resistance.

The goals of this analysis are threefold: (1) to provide a comprehensive review of current video-call-based censorship resistance techniques, (2) to assess their strengths, weaknesses, and practical applications, and (3) to offer recommendations for future research and tool development. This study aims to support both end users seeking secure communication methods and developers working to enhance censorship circumvention technologies.

2 Background

2.1 Basic concepts

- WebRTC **(Web Real-Time Communication)**: is a set of open source technologies that enables real-time communication through browsers and mobile applications, facilitating the exchange of multimedia content without the need for additional plugins. Its popularity in applications such as Facebook Messenger and Discord makes it difficult for censors to block WebRTC without affecting these legitimate services, making it an effective tool for circumventing censorship. [10, p. 15]
- **Tunneling:** Tunneling is a technique that allows one network protocol to be encapsulated within another. This is used to create a secure and private communication channel over a public network, such as the internet. In the context of censorship circumvention, tunneling is used to hide traffic from censors, making it appear as normal and legitimate traffic. For example, Protozoa uses WebRTC tunneling to encapsulate data within video streams, making detection by censors more difficult.[3, 12]
- **Steganography:** Steganography is the practice of hiding information within other types of data, such as images, videos, audio files, or even text. Unlike cryptography, which encrypts information to make it unreadable, steganography aims to hide the very existence of the information. In the context of censorship circumvention, steganography is used to covertly transmit information, preventing censors from detecting it[10, p. 8-14].
- **Protocol Obfuscation:** Protocol obfuscation is a technique that aims to modify the behavior of a network protocol to make it harder for censors to detect and block. This can be achieved by modifying packet headers, using non-standard ports, fragmenting data, or introducing random noise into the communication.[6, 3, 9]
- **Proxies in Decentralized Networks:** Proxies are servers that act as intermediaries between clients and other servers, enabling access to blocked content while masking the user's IP address and the true location of the destination server. When integrated into decentralized networks, proxies gain additional significance, as these networks lack a single point of control, making them inherently more resistant to censorship. Decentralized architectures distribute the responsibility and operation of proxies across multiple nodes, reducing the risk of a single point of failure. This design ensures

that even if some proxies are compromised or blocked, the network can continue functioning, making it a vital component in censorship circumvention efforts.

- **Sybil Attack:** A Sybil attack is a type of attack where an attacker creates multiple fake identities (Sybil nodes) to infiltrate and influence the behavior of a network. In censorship circumvention protocols, such attacks pose significant risks by allowing censors to compromise decentralized networks. By gaining control of a sufficient number of Sybil nodes, an attacker can manipulate traffic routing to intercept or block data, spread false information to mislead users, launch denial-of-service attacks against critical nodes, compile blacklists of users attempting to bypass censorship, and erode trust in the network. These actions can severely undermine the effectiveness of the protocol, making resistance to Sybil attacks a critical design consideration for any censorship circumvention system [2, p. 41].

- **Correlation Attacks and Traffic Analysis:** Correlation attacks are advanced techniques used to identify and track users in a network, even when they employ anonymization protocols. These attacks leverage traffic analysis to examine patterns and characteristics of network traffic, such as packet timings and sizes, to correlate flows observed at different network points. In the context of internet censorship, censors use correlation attacks to deanonymize users by linking their incoming and outgoing traffic, thereby compromising their privacy and security. There are two primary types of correlation attacks[11, p. 1491]:
  - **Passive Correlation Attacks**, where attackers observe traffic patterns without interfering, and later analyze the data to match flows and identify users. [11, p. 1491]
  - **Active Correlation Attacks**, where attackers inject identifiable markers or watermarks into the traffic to trace its origin or confirm its presence in another location.[11, p. 1491]

- **Deep Packet Inspection (DPI)**: is an advanced network traffic analysis technique that examines the content of data packets, rather than just header information such as IP addresses and port numbers. DPI allows censors to identify and block traffic based on its content, such as specific protocols, applications or even keywords within the message content.

2.2 Characteristics of an Ideal System for Accessing Information in a Censored Network

In a censored network, an ideal system for accessing information and communicating freely should have a series of features that allow it to overcome restrictions imposed by censors while also protecting the privacy and security of users.

- **Unblockable:** The system must be resistant to blocking attempts by censors. This involves using techniques that make it difficult to identify and block traffic, such as:
  - **Traffic obfuscation:** Communications should be indistinguishable from legitimate traffic, using protocols and masking techniques that avoid detection through pattern or signature analysis.[11, 12]
  - **Decentralization:** A decentralized architecture makes censorship difficult because there is no single point of failure that can be blocked. [2]
  - **Covert channels:** Encapsulating traffic within video streams, online games, or other multimedia protocols can provide an additional layer of resistance to blocking.[6, 12, 7]
  - **Use of proxies:** Proxies, especially those located in uncensored networks, can act as intermediaries to access blocked content and make communication tracking more difficult. [11, 5, 3]
  - **Resistance to Sybil attacks:** The system must be able to identify and mitigate the influence of fake nodes (Sybil) introduced by censors to compromise the network. [2]

- **Unobservable:** The system should be able to hide its operation from censors, avoiding detection through traffic analysis. Some strategies to achieve this are:
  - **Traffic mimicry:** The system should generate traffic that closely resembles legitimate network traffic, using similar communication parameters and patterns. [12]
  - **Steganography:** Steganography, as used in Stegozoa[6], allows information to be hidden within other data, such as images or videos, making it difficult to detect the covert channel.
  - **Use of K-anonymization techniques:** TorKameleon uses K-anonymization to fragment and redirect traffic through multiple proxies, making it difficult to track the user and correlate their traffic. [11]

- **Resistant to Correlation Attacks:** Ideal systems must be able to resist attacks that attempt to correlate incoming and outgoing network traffic to identify users and communication patterns. This involves:
  - **Multipath routing:** Using multiple paths to send and receive data makes it harder for censors to correlate traffic. [11]
  - **Traffic pattern variation:** Dynamically changing communication patterns, such as packet sizes and time intervals, makes it harder to identify unique patterns.[11]
  - **Network layer anonymization:** Use anonymization techniques, like Tor, to hide the user's IP address and make tracking online activity more difficult.

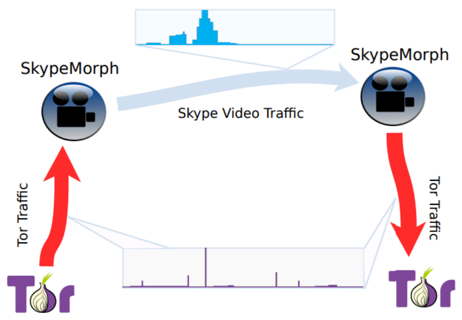

Figure 1: Overview of the SkypeMorph Architecture. The histograms show the distribution of packet sizes in Tor (at the bottom) and Skype video (at the top).[9]

- **Scalable:** The system must be capable of handling a large number of users and connections without compromising performance or security. [2]
- **Easy to Use:** Usability is crucial for the widespread adoption of the system. It should be easy to install, configure, and use, even for users without advanced technical knowledge.
- **Flexible:** The system should be adaptable to different types of censorship and network environments.
- **Secure:** The system must protect the privacy and security of users, including the confidentiality of their communications, and data integrity.

3 Summary of most relevant methods

In order to better understand the methods of circumventing censorship, we will proceed to a summary of those considered most relevant, ordered according to their complexity and evolution.

3.1 SkypeMorph

3.1.1 Obfuscation Technique

SkypeMorph employs a technique known as *protocol mimicry*. Its goal is to disguise Tor network traffic, making it appear as a Skype video call. This technique relies on altering the traffic to mimic the characteristics of legitimate Skype traffic.

To achieve this, SkypeMorph manipulates elements such as packet sizes and timing intervals to match typical patterns of Skype video calls. The core idea is that, since Skype is a widely used platform, blocking all of its traffic to detect SkypeMorph users would result in unacceptable collateral damage for the censor.[9]

3.1.2 Internal Operation

The SkypeMorph process is divided into two phases: **Configuration Phase**

- **Login and Call Setup:** Both the client and the bridge log in to the Skype network using the Skype API with unique credentials. The bridge listens for incoming calls while the client prepares to initiate a connection.
- **Key Exchange:** The client generates a public key and sends it to the bridge via a Skype text message, along with its IP address and selected UDP port. The bridge responds with its own public key and port. Both parties compute a shared secret key and verify it by exchanging hashes. If verification succeeds, the client initiates a Skype video call to the bridge.
- **TCP and UDP Connection Maintenance:** The video call ends after a brief period, but the underlying UDP connection is maintained. The bridge dynamically selects a new UDP port for subsequent connections to mimic typical Skype behavior.

Traffic Shaping Phase

- **Call Termination:** After the setup phase, the video call is terminated, but the UDP connection remains active.
- **Data Tunneling:** Tor traffic is then sent through this UDP connection, disguised as Skype traffic.
- **Traffic Shaping:** A *traffic-shaping oracle* modifies Tor packets to imitate Skype traffic patterns using *naive traffic shaping* or *enhanced traffic morphing*.
- **Encryption and Authentication:** SkypeMorph adds an additional AES encryption layer and HMAC authentication to the already encrypted Tor traffic.

3.1.3 Resource Consumption/Performance

- **Bandwidth Overhead:** SkypeMorph introduces bandwidth overhead because it must transmit the same volume of data as a legitimate Skype video call, even when there is insufficient Tor traffic to fill the packets.
- **Download Speed:** Tests show an average download speed of 34 KB/s with a 28% overhead, which is comparable to standard Tor connections .
- **CPU and Memory Usage:** CPU and memory consumption is moderate, indicating that SkypeMorph can run on common hardware platforms.

[9, 3]

3.1.4 Censor Resistance and Vulnerabilities

- **Strength Against Censorship:** SkypeMorph's primary strength is its ability to imitate Skype traffic, making detection and blocking difficult for censors. To block SkypeMorph, a censor would need to block Skype entirely or identify the presence of the software on a remote node [9].

- **Bridge Enumeration Attack:** A censor could run their own SkypeMorph bridges and distribute them to users, allowing the detection of connections to the bridge, though not the Tor traffic itself [9].

- **Traffic Analysis Vulnerability:** Sophisticated traffic analysis attacks that examine higher-order statistics may be capable of distinguishing SkypeMorph traffic from legitimate Skype traffic [9].

- **Incomplete Skype mimicry:** SkypeMorph does not mimic all aspects of Skype, such as HTTP update and login traffic, the Skype TCP channel, and SoM fields in Skype UDP packets. This allows censors to distinguish SkypeMorph from genuine Skype, even with passive and low-cost attacks [8, p. 70].

- **Reuse of Pre-recorded Skype Traces:** SkypeMorph rely on pre-recorded traffic traces to imitate Skype's behavior. Censors could compare observed flows with these patterns to detect imitation. [8, p. 71].

- **Inability to Mimic Supernode Behavior:** Skype uses supernodes to transmit multimedia traffic and signaling information for ordinary clients. SkypeMorph cannot function as genuine supernodes, enabling detection through proactive attacks[8, p. 71].

- **Inadequate Reaction to Errors and Network Conditions:** SkypeMorph+ and StegoTorus+ cannot accurately mimic the complex dynamic dependencies between network conditions and Skype's control traffic. Censors could exploit this by manipulating connections and observing endpoint reactions.[8, p. 72].

- **Inability to Mimic Implementation-Specific Artifacts:** SkypeMorph cannot replicate Skype's implementation-specific features, such as version-specific quirks and errors. Censors can detect SkypeMorph using packet inspection or fingerprinting techniques[8, p. 70].

In summary, **the partial imitation of Skype by SkypeMorph makes it detectable even by weak censors**. Its attempts to mimic Skype's traffic patterns and behavior are not sufficiently convincing to evade censorship.

3.1.5 Ease of Use

Using SkypeMorph can be challenging for average users due to its relatively complex deployment and configuration requirements [4].

3.2 CensorSpoofer

3.2.1 Obfuscation Technique

CensorSpoofer decouples the upload and download channels, using a low-bandwidth indirect channel to deliver upload messages (URLs) and a high-bandwidth direct channel to download web content. The upload channel hides request

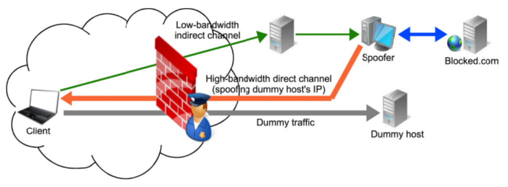
Low-bandwidth indirect channel
Spoofer
Blocked.com
High-bandwidth direct channel (spoofing dummy host's IP)
Client
Dummy traffic
Dummy host

Figure 2: The CensorSpoofer framework. The user pretends to communicate with an external dummy host legitimately, and sends URLs to the spoofer via a low-bandwidth indirect channel (e.g., steganographic IM/Email). The spoofer fetches blocked webpages, and injects censored data into the downstream flow towards the user by spoofing the dummy host's IP. [12]

content via steganographic encoding within email or instant messages, while the download channel uses IP address spoofing so that the real address of the proxy is not revealed to either legitimate users or censors [12].

3.2.2 Internal Operation

Download Channel: CensorSpoofer employs IP spoofing to send data from the proxy to the user without revealing the actual origin. This spoofed channel supports one-way communication. To avoid detection by censors, CensorSpoofer tunnels download data through a spoofed encrypted VoIP session, since VoIP protocols do not require tight endpoint synchronization and conceal content from censors [12].

- **Decoy Host Selection:** To covertly transmit censored data, CensorSpoofer uses a decoy host as a cover. Each decoy host must appear legitimate to censors; for example, if the cover session is a VoIP call, the host's VoIP port should appear open [12].

- **AS Path Masking:** To prevent censors from detecting anomalies in the decoy host's AS path, CensorSpoofer uses *traceroute* to determine the real AS path from the spoofer to the client. It then selects a decoy host in an AS adjacent to the spoofer's AS in the reference path [12].

Upload Channel: Each user sends upload messages through a steganographic channel embedded in indirect communications, such as instant messaging or email. The steganographic process uses a secret encoding key to remain undetectable. To resist insider attacks, each user registers a separate key pair upon joining the system [12].

- **Unique IDs and Invitation Process:** To prevent censors from identifying legitimate users via SIP IDs and upload IDs, CensorSpoofer assigns each client a unique SIP ID and upload ID. Instead of the spoofer generating these IDs, each client registers a SIP ID and upload ID on behalf of the spoofer and delivers them through an invitation process [12].

3.2.3 Resource Consumption/Performance

Experimental results show that clients successfully downloaded the blocked Wikipedia page (in China) using CensorSpoofer. With the G.711 or G.722-64 codec, downloading the entire page took 27 seconds, while loading the HTML file alone took 6 seconds. Compared to Tor and a NetShade4-based proxy, CensorSpoofer took longer to download pages due to the need to reshape download traffic into low-bandwidth VoIP traffic. However, download times for small web content, such as HTML files, remain acceptable [12].

3.2.4 Censorship Resistance and Vulnerabilities

- **Traffic Analysis Vulnerability:** Advanced traffic analysis can detect inconsistencies in VoIP traffic patterns, revealing the use of evasion systems like CensorSpoofer [12].

- **Manipulation of the Tag Field:** SIP messages in CensorSpoofer use a hash of the spoofed IP address as a tag. Censors can exploit this by altering the tag field, causing CensorSpoofer to terminate the call, unlike genuine SIP clients that continue the session[8, p. 74]

- **SIP Probing:** Censors can send SIP messages to spoofed IP addresses, probing for responses to distinguish between CensorSpoofer recipients and genuine clients [12].

- **Upstream Packet Manipulation:** Dropping RTP packets does not elicit the expected reactions from CensorSpoofer, making it distinguishable from genuine RTP sessions.[8, p. 75]

Conclusion: While CensorSpoofer incorporates mechanisms to mimic legitimate SIP and VoIP traffic, its limitations in handling traffic manipulation, SIP probing, and reaction to packet drops make it vulnerable to detection by modern censors. In contemporary censorship scenarios, these vulnerabilities render CensorSpoofer less effective compared to more robust evasion systems.

3.2.5 Ease of Use

To use CensorSpoofer, a user initiates (or pretends to initiate) a legitimate communication session (e.g., a VoIP call) with a decoy host outside the censor's network. The proxy (spoofer) injects censored data into the download stream by spoofing the IP address of the decoy host, making it appear as if the user is communicating solely with the decoy. Meanwhile, the user sends URLs to the spoofer through a low-bandwidth indirect channel, such as steganographic instant messaging or email. Implementing and using CensorSpoofer requires advanced configuration and setup [12].

3.3 Protozoa

3.3.1 Obfuscation Technique

Protozoa conceals transmitted information by **replacing the data in encoded video frames with the payload of IP packets**. This replacement occurs right after the video codec applies lossy compression, modifying the WebRTC stack of the Chromium browser component in Protozoa [3, p. 36].

3.3.2 Internal Operation

Protozoa works by intercepting the WebRTC video data stream and replacing the encoded video frame content with covert data. This process is performed by inserting **two hooks** in the WebRTC software stack: one upstream and one downstream.

- The **upstream hook** intercepts frame data before it is sent to the network.

- The **downstream hook** intercepts frame data arriving from the network.

The following steps describe the data encapsulation process:

- Protozoa encapsulates IP packets into Protozoa messages, which are segmented to fit within the available space in video frames.

- The Protozoa messages are inserted into the *Encoded Frame Bitstream Partitions* (EFBP) field of SRTP packets carrying video data.

- At the receiving end, Protozoa extracts the Protozoa messages from the SRTP packets, reassembles the IP packets, and sends them to their destination.

3.3.3 Resource Consumption/Performance

Protozoa has demonstrated a covert channel throughput of **1.4 Mbps** and a channel efficiency of 98.8% under normal network conditions. This allows the use of common Internet applications, such as web browsing or bulk data transfer. In lab tests, using a video resolution of 640x480, an average throughput of 1422 Kbps was achieved. The client and proxy VMs peaked at a CPU usage of 21.6% and memory usage of 596 MB, indicating that Protozoa can run on various hardware platforms [3, p. 42-43].

3.3.4 Censorship Resistance and Vulnerabilities

Protozoa is designed to resist machine-learning-based traffic analysis by state-level adversaries [3, p. 36]. However, it presents several vulnerabilities that limit its effectiveness in modern censorship scenarios:

- **Man-in-the-Middle Attacks:** Protozoa is vulnerable to MitM attacks, particularly when communication is mediated by WebRTC gateways controlled by adversaries. This enables interception and manipulation of traffic [3, p. 39][6, p. 1154].

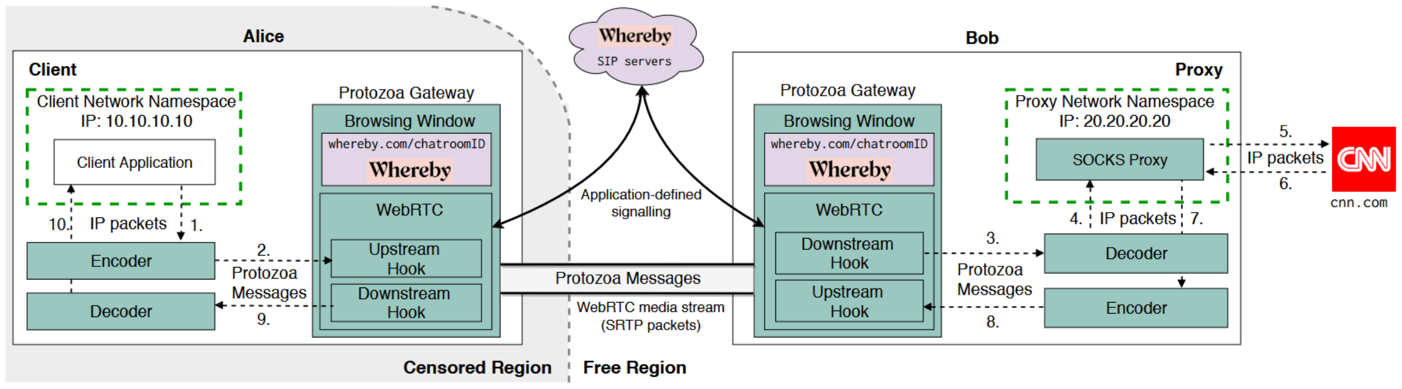

Figure 3: Architecture of Protozoa: The components of the system are highlighted in a darker shade.[3, p. 38]

- **Lack of resistance to state-level attacks:** Protozoa cannot cope with a number of state-level attacks, such as the deployment of (fake) Sybil users or the use of adversary-controlled WebRTC broadcast applications. [2, p. 37-38]

- **Dependence on Peer-to-Peer Connections:** Protozoa assumes peer-to-peer data exchange, a condition not always met as WebRTC applications often use centralized gateways. This creates opportunities for censorship via traffic inspection or blocking.[6, p. 1154]

- **Pattern-Based Detection:** Long-term profiling of WebRTC connection patterns can expose unusual interactions associated with Protozoa's covert channels [3, p. 46].

- **Susceptibility to Correlation Attacks:** Protozoa is untested against active correlation attacks, such as injecting delays or other markers into network segments to trace covert traffic.[11, p. 1490]

Conclusion: Protozoa demonstrates innovative strategies for evading censorship but is hampered by vulnerabilities that limit its resilience against state-level adversaries. Its reliance on peer-to-peer architectures, lack of robust mechanisms against traffic manipulation, and susceptibility to direct content inspection and active correlation attacks make it less secure for long-term use in modern, highly monitored environments.

3.3.5 Ease of Use

Deploying Protozoa can pose challenges, necessitating users to compile the Chromium browser, and it lacks compatibility with Tor [11, p. 1491].

3.4 Stegozoa

3.4.1 Obfuscation Technique

Stegozoa, presented as an improvement in the security of Protozoa, hides transmitted information within video frames of WebRTC calls. It leverages a combination of **Syndrome-Trellis Coding (STC)**[1], a state-of-the-art adaptive steganographic technique, and data encoding within the least significant bits of the Quantized Discrete Cosine Transform (QDCT)[2] coefficients of residual video frames [6] [6].

3.4.2 Internal Operation

Stegozoa operates as a library (*libstegozoa*) that integrates with WebRTC applications. This library intercepts video frames before they are encoded and sent by the application, embedding covert data using the selected steganographic technique. At the receiving end, the library extracts the covert data from the received video frames.

To facilitate message exchange, Stegozoa implements a message fragmentation and reassembly protocol that allows breaking messages into smaller packets and efficiently transmitting them through the covert channel. Additionally, Stegozoa includes a retransmission mechanism to ensure reliable message delivery in case of packet loss [6].

3.4.3 Resource Consumption/Performance

Stegozoa's performance depends on the steganographic embedding capacity. Higher capacity yields greater throughput but reduces resistance to analysis. In lab tests, Stegozoa achieved a throughput of **11.4 Kbps** with a high embedding capacity ($\alpha = 0.50$). While this is unsuitable for bulk data transfer or live video streaming, it supports the exchange of

[1]

[2]QDCT stands for Quantized Discrete Cosine Transform Coefficients, which refers to the coefficients obtained by applying the Discrete Cosine Transform (DCT) to a signal, such as an image or video frame, and then subjecting them to a quantization process. DCT decomposes the signal into cosine waves of different frequencies, and quantization reduces the number of bits needed to represent them, which is achieved by dividing the coefficients by a scale factor and rounding the results. QDCT coefficients are fundamental in video and audio compression, such as in the VP8 codec used by WebRTC, and are also used in steganography to hide information. In this context, QDCT coefficients are slightly modified to embed data without the changes being perceptible to the human eye, allowing secret messages to be hidden within video streams without causing visible distortion, as the Stegozoa tool does[10, p. 18-20].

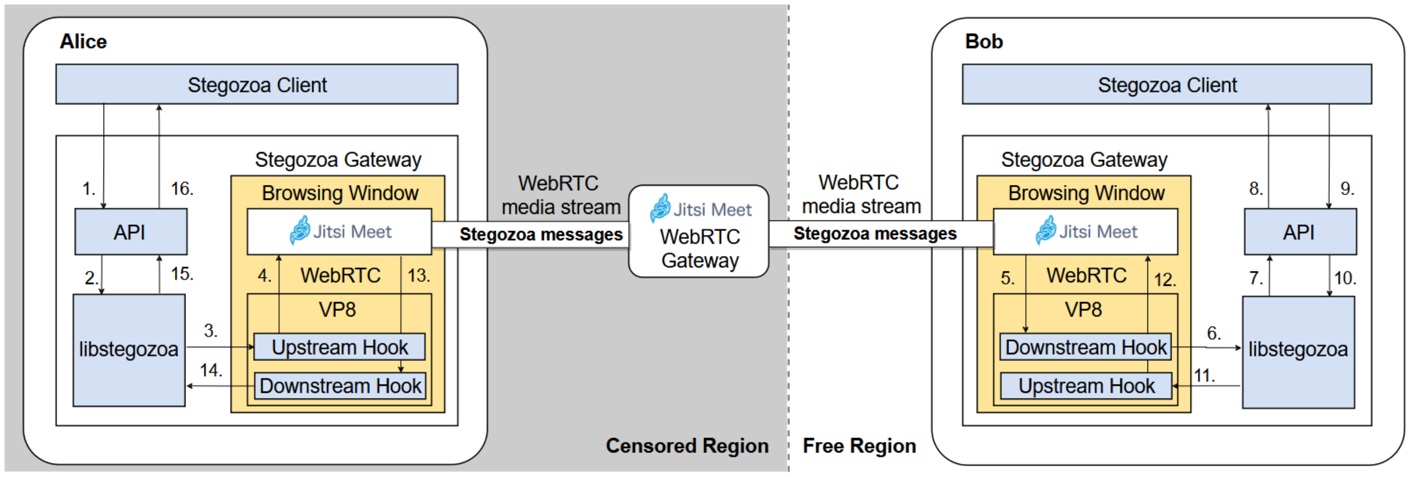

Figure 4: Architecture of Stegozoa: The components of the system are highlighted in blue[6]

small content such as **Twitter feeds** or **Wikipedia articles** [6].
The embedding parameter α determines the fraction of embedding capacity used:

- A **higher** α embeds more data, increasing covert channel throughput but also introducing more video distortion, making it more susceptible to steganalysis [6].
- A **lower** α embeds less data, reducing throughput but enhancing security by making the covert channel harder to detect [6].

3.4.4 Censorship Resistance and Vulnerabilities

Stegozoa is a censorship evasion system based on WebRTC that uses steganography to embed covert data within video media exchanged during WebRTC calls. Designed to ensure end-to-end security even in the presence of potentially malicious WebRTC gateways, Stegozoa demonstrates strong resistance to traffic analysis and steganalysis, even under adverse network conditions like packet loss[6].. However, it exhibits several vulnerabilities:

- **Vulnerability to Media Manipulation:** Stegozoa is not designed to resist adversaries who can directly manipulate encoded video streams, potentially disrupting the identification of coefficients carrying covert data or altering the embedded message[10, p. 26].
- **Fragility to Recoding:** Routine operations like video transcoding by WebRTC gateways can strip or corrupt the covert channel, breaking the communication[10, p. 3].
- **Embedding Space Fragility:** The dynamic filtering of coefficients with zero or one values during embedding leads to inconsistencies post-recoding, hindering message recovery[10, p. 28]..
- **Lack of Error Correction:** Stegozoa does not implement error correction mechanisms at the stego vector level, making it vulnerable to transmission modifications[10, p. 29]..

Proposed Mitigations:

- Employing static embedding spaces known to both sender and receiver to enhance robustness against recoding[10, p. 47].
- Integrating error correction codes to recover messages post-recoding [10, p. 49].
- Introducing a message control layer to manage retransmissions and handle potential corruption[10, p. 53].
- Opting for embedding in keyframes or all frames, balancing between robustness and efficiency[10, p. 55].
- Combining Syndrome-Trellis Coding (STC) with error correction schemes to improve resilience[10, p. 56].

Conclusion: Stegozoa improves upon Protozoa by incorporating advanced steganographic techniques, offering significant resistance to traffic analysis and steganalysis. However, its vulnerabilities to video manipulation and recoding highlight areas needing enhancement. Implementing the proposed mitigations could strengthen its robustness, ensuring greater security and efficacy in modern censorship evasion scenarios.

3.4.5 Ease of Use

Stegozoa integrates as a library within WebRTC applications, simplifying usage for end users. Users only need to **initiate a video call** with a trusted contact outside the censored region, and Stegozoa will **tunnel traffic** through the call. It has also been tested with various calling services [6].

3.5 TorKameleon

3.5.1 Obfuscation Technique

TorKameleon is a censorship evasion system that combines K-anonymization[3] techniques and traffic encapsulation in covert channels, such as WebRTC video calls or TLS tunnels, to improve Tor's resistance against passive and active correlation attacks. It fragments user traffic, mixes it with that of other users, and routes it through a distributed network of proxies, making it difficult for censors to detect. Upon reaching the final proxy, the traffic is decapsulated and sent to the Tor network.[11].

3.5.2 Internal Operation

TorKameleon implements K-anonymization by splitting user traffic and forwarding it through multiple proxies within a pre-Tor network. This allows one user's traffic to be mixed with that of additional KK users, reducing the probability of correlation to 1/K. TorKameleon can operate in three distinct modes:

- **Pluggable Transport Mode:** In this mode, TorKameleon functions as a pluggable transport for Tor. This allows users to connect to the Tor network via a TorKameleon bridge, hiding their traffic within WebRTC video calls or TLS tunnels.
- **Standalone Mode:** Users can deploy a network of TorKameleon proxies to route their traffic. The information is fragmented and sent through different proxies, making it harder for censors to correlate traffic [11].
- **Combined Mode:** This mode combines the previous two. User traffic is routed through a network of proxies and then sent to the Tor network via a TorKameleon bridge [11].

Regardless of the operating mode, TorKameleon functions as follows:

1. **Route Establishment:** The user configures a proxy route, or the TorKameleon gateway software determines it automatically.
2. **Proxy Connection:** The user connects to the first proxy in the route.
3. **Encapsulation and Forwarding:** The proxy encapsulates the user's traffic in a covert channel (WebRTC or TLS) and sends it to the next proxy in the route.
4. **Repetition of Steps:** Steps 2 and 3 are repeated until the traffic reaches the final proxy.
5. **Connection to Tor:** The final proxy sends the traffic to the Tor network via the TorKameleon bridge.

[3]K-anonymization is a data anonymization technique that seeks to protect individuals privacy by ensuring that no record in a dataset can be uniquely identified. This is achieved by generalizing data until each record is indistinguishable from at least k-1 other records in the dataset.[11]

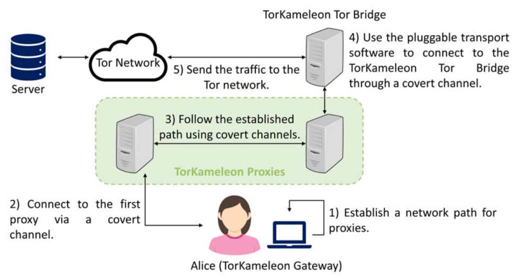
TorKameleon Tor Bridge
4) Use the pluggable transport software to connect to the TorKameleon Tor Bridge through a covert channel.
Server
Tor Network
5) Send the traffic to the Tor network.
3) Follow the established path using covert channels.
TorKameleon Proxies
2) Connect to the first proxy via a covert channel.
1) Establish a network path for proxies.
Alice (TorKameleon Gateway)

Figure 5: System Model and Workflow of the TorKameleon Ecosystem. When using the pluggable transport without proxies, the user establishes a direct connection to the TorKameleon Tor Bridge through the TorKameleon pluggable transport client-side, which operates on the user's local device.[11]

3.5.3 Resource Consumption/Performance

TorKameleon's performance is comparable to other WebRTC-based censorship evasion systems. In tests, TorKameleon achieved performance similar to a regular Tor connection, with latencies between **530–655 ms** when using TLS encapsulation [11]. When using WebRTC encapsulation, performance decreased as more users connected to the TorKameleon bridge. Further studies are needed to accurately determine TorKameleon's resource consumption [11].

3.5.4 Censorship Resistance and Known Vulnerabilities

TorKameleon has been tested against both passive and active correlation attacks. Results show that it is resistant to passive correlation attacks, meaning censors cannot easily identify TorKameleon traffic based solely on traffic analysis. Additionally, TorKameleon demonstrates resistance to active correlation attacks using watermarking techniques, provided small data blocks are used. However, certain configurations and operational parameters can introduce vulnerabilities:

- **Data Block Size:** The size of the data block used to encapsulate traffic within WebRTC video frames significantly impacts TorKameleon's detectability. Larger data blocks (e.g., over 1050 bytes) increase the likelihood of detection during active correlation attacks, as they provide more space for censors to inject and detect watermarks.[11, p. 1495]
- **WebRTC Encapsulation Mode:** TorKameleon supports two encapsulation modes: *ADD* and *REPLACE*. While the *ADD* mode offers higher performance, it is more vulnerable to active correlation attacks. The *REPLACE* mode, which replaces video frame content with data blocks, has shown greater resistance to watermark detection, especially with larger data blocks.[11, p. 1495]
- **Number of TorKameleon Proxies:** The effectiveness of TorKameleon relies on a sufficiently large network of proxies. A limited number of proxies (less

than four) increases the risk of user de-anonymization if a censor monitors traffic from a significant portion of these proxies. [11, p. 1494]

- **Potential Undetected Vulnerabilities:** As TorKameleon is a relatively new system with limited academic scrutiny (only five citations as of December 2024), it is possible that additional vulnerabilities remain unidentified, particularly against sophisticated state-level adversaries.

Conclusion: While TorKameleon shows promising resistance against both passive and active correlation attacks under specific configurations, its security is highly dependent on proper setup and operational parameters. Vulnerabilities related to block size and encapsulation modes should be carefully mitigated. Additionally, expanding the network of proxies and ensuring geographic diversity could further enhance its resilience. Given its novelty and limited peer review, ongoing research and real-world testing are essential to identify and address any undiscovered weaknesses.

3.5.5 Ease of Use for End Users

The current implementation of TorKameleon is a prototype designed as a proof of concept rather than a tool intended for end users. It lacks integration with web browsers or user-friendly interfaces, focusing instead on demonstrating the system's technical feasibility. According to its publicly available repository [1], users must manually configure various components, including WebRTC parameters, the STUN/TURN server[4], and the TorKameleon proxy. The setup process also involves the use of advanced tools like Docker and requires familiarity with network configurations, such as port forwarding. These aspects make the prototype unsuitable for average users without significant technical expertise.

3.6 Snowflake

3.6.1 Obfuscation Method

Snowflake does not rely on traditional information-hiding techniques such as steganography or packet manipulation to mask transmitted data. Its censorship resistance derives from the use of a **vast, constantly changing network of temporary proxies** known as "snowflakes." These proxies, run by volunteers through web browsers, act as intermediaries, relaying traffic from censored clients via peer-to-peer WebRTC protocols to a centralized bridge. The information itself is not hidden; instead, Snowflake's strategy focuses on making communication harder to block by avoiding fixed censorship targets [5].

[4]STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT) servers are crucial for establishing WebRTC connections over NAT networks. STUN servers allow clients to discover their public IP address and the port assigned by their NAT router, facilitating direct P2P connections. If a direct connection is not possible, TURN servers act as intermediaries, relaying traffic between clients. While STUN servers allow direct connections when conditions permit, TURN servers provide a solution when such connections are not feasible due to NAT restrictions.

3.6.2 Internal Operation

1. **Discovery:** The client, located in a censored region, establishes a connection with a centralized Snowflake bridge using various discovery techniques, such as domain fronting, AMP caching, and Amazon SQS services. This dynamic discovery process circumvents blocking by leveraging third-party services that are difficult to block without significant collateral damage [5].

2. **Proxy Connection:** After connecting to the bridge, the client receives a list of available Snowflake proxies. A peer-to-peer connection with one of these proxies is then established via WebRTC [5].

3. **Data Transfer:** The Snowflake proxy acts as a simple data relay, copying information between the client and the bridge without performing complex processing [5].

4. **Protocol Layers:** Communication occurs through a protocol stack that encapsulates data, including WebRTC, WebSocket, TLS, and a custom Snowflake protocol. This layered structure provides flexibility and security to the connection [5].

5. **Integration with Tor:** The Snowflake bridge connects to the Tor network, enabling clients to access Tor and browse the Internet anonymously. Combining Snowflake with Tor offers an additional layer of protection against censorship and surveillance [5].

3.6.3 Resource Consumption/Performance

Although the authors do not provide precise details on Snowflake's resource consumption and performance, it is described as an **ultralightweight system**. Since proxies are implemented in JavaScript and run in web browsers, the resource consumption is relatively low. Snowflake has shown the ability to handle a large number of clients and high bandwidth during network disruptions, indicating satisfactory performance [5].

3.6.4 Censorship Resistance and Known Vulnerabilities

Resistance: Snowflake's censorship resistance is rooted in the dynamic and distributed nature of its proxy network. The protocol leverages a large and constantly changing pool of volunteer proxies, with short lifespans and rapid rotation of IP addresses, which makes it challenging for censors to identify and block traffic effectively. Furthermore, Snowflake employs discovery techniques based on third-party services, enhancing its resilience against blocking attempts [5].

Vulnerabilities: Despite its strengths, Snowflake presents several vulnerabilities, particularly due to its reliance on WebRTC and specific implementation details:

- **Proxy Enumeration:** Censors could attempt to systematically identify and block all available Snowflake proxies. While Snowflake's distributed design mitigates

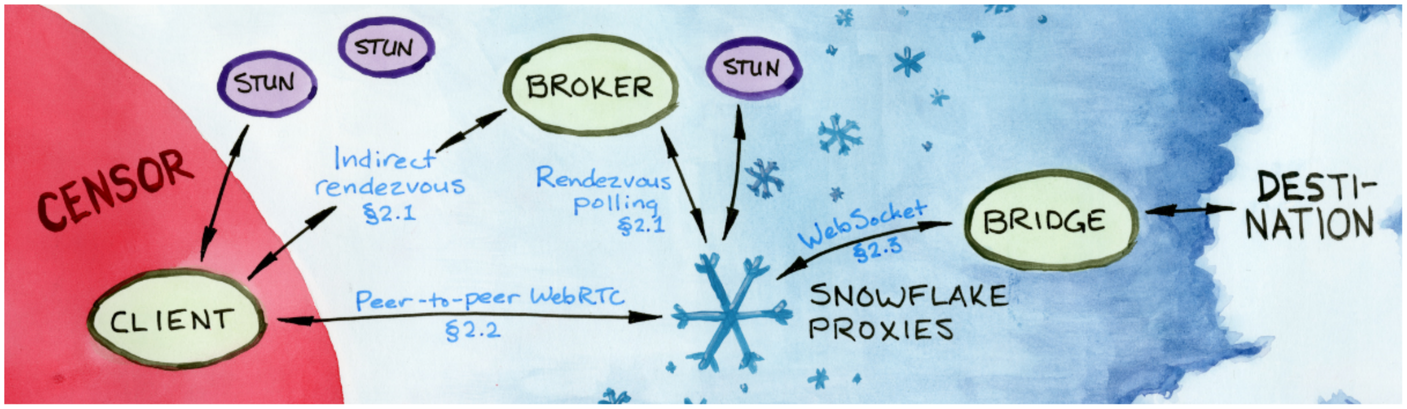

Figure 6: Architecture of Snowflake. The client contacts the broker through an indirect rendezvous channel with high blocking resistance. The broker matches the client with one of the proxies that are currently polling. The client and proxy connect to one another using WebRTC. The proxy connects to the bridge, then begins copying traffic in both directions. If the proxy disappears, the client does another rendezvous and resumes its session with a new proxy.[5]

this threat to some extent, achieving an optimal balance between security, usability, and decentralization remains a challenge [5].

- **DTLS Handshake Fingerprints:** The DTLS handshake phase in Snowflake's WebRTC implementation has been identified as a source of unique fingerprints, which can be exploited for traffic identification. These fingerprints demonstrate high discriminatory power, enabling adversaries to differentiate Snowflake traffic from other WebRTC-based applications. However, developers can counteract this vulnerability by obfuscating or transforming the DTLS handshake [5].

- **TLS Fingerprint Blocking:** Snowflake uses uTLS to obfuscate TLS fingerprints, but incidents like the blocking of specific TLS fingerprints in Iran during 2022 underscore the need for agile and diverse fingerprint management [5].

Conclusion: Snowflake demonstrates a robust approach to censorship resistance through its distributed and adaptive proxy network. However, the protocol's reliance on WebRTC and the specific vulnerabilities it introduces, specially fingerprint patterns, highlight areas requiring ongoing attention. Addressing these vulnerabilities with enhanced obfuscation techniques and dynamic fingerprint management is essential to maintaining Snowflake's efficacy in circumventing censorship.

3.6.5 Ease of Use for End Users

Snowflake is designed for a seamless user experience, requiring no special configurations or technical knowledge from the client. The client software automatically and transparently handles bridge connections, proxy selection, and data transfer, offering an experience similar to other web applications [5].

4 Summary comparison mentioned methods

Table 1: Comparison of Censorship Evasion Protocols

Protocol	**Obfuscation Technique**	**Internal Operation**	**Usable for Browsing?**	**Security**	**User-Friendly?**
SkypeMorph	Mimics Skype traffic patterns	Alters packet sizes and timing to resemble a Skype call.	No, bandwidth overhead (28%) and 34 KB/s speed limit comfortable browsing.	Vulnerable to detailed traffic analysis, partial Skype mimicry. Not secure for real-world use.	No, difficult to set up for average users.
CensorSpoofer	VoIP encapsulation with IP spoofing	Download via VoIP; upload via steganographic IM/email.	No, VoIP format causes delays (27s for a page).	Vulnerable to SIP traffic manipulation and analysis. Not usable in real scenarios.	No, requires advanced configuration.
Protozoa	Replaces WebRTC video frame content	Inserts hidden data into encoded WebRTC frames.	Yes, 1.4 Mbps throughput allows basic browsing.	Vulnerable to Man-in-the-Middle and correlation attacks. Not secure against state-level adversaries.	No, difficult to implement; requires compiling Chromium.
Stegozoa	Steganography in video frame QDCT coefficients	Hides data in least significant bits of video frames.	No, low throughput (11.4 Kbps) limits browsing.	Vulnerable to video recoding and media manipulation. Moderate security in real-world use.	Yes, relatively easy to use as a library integration.
TorKameleon	Tor traffic encapsulation in WebRTC/TLS.	Hides traffic within WebRTC video or TLS tunnels.	More or less, latency (530-655 ms) limits usability.	Vulnerable if few proxies are used; Potential undetected vulnerabilities. Potentially secure	No, complex prototype unsuitable for average users.
Snowflake	Distributed temporary WebRTC proxies	Dynamic connection to volunteer proxies via a central bridge.	Yes, lightweight and efficient for browsing.	Vulnerable to DTLS fingerprinting. Generally secure with ongoing updates.	Yes, very easy to use; automatic for end users.

Answers to Key Questions

User-Oriented Considerations

Answers to Key Questions

User-Oriented Considerations

1. **What factors should end users consider when selecting a censorship-resistant system, given their technical proficiency and available resources?**

   When selecting a censorship-resistant system, users should consider their technical skills, the capabilities of the censor, and their specific needs. The system should be easy to use, resistant to blocking, and meet performance requirements. The following factors are crucial:

   - **Ease of use**: For non-technical users, simplicity is key. Systems that require minimal setup and function like common web applications are ideal. `Snowflake` **is the only modern system that offers true accessibility for end users**, as it runs through browser extensions with no configuration required.
   - **Resilience to blocking**: The system should resist common censorship techniques such as IP blocking, DNS tampering, and deep packet inspection. Techniques like traffic obfuscation and dynamic proxy rotation enhance resilience. `Snowflake` **uses temporary, ever-changing proxies, making it difficult for censors to block.**
   - **Performance**: Users should consider whether the system meets their speed and latency needs. Systems that use steganography or tunneling may have slower performance. For basic web browsing, `Stegozoa` and `Snowflake` offer reliable performance for low to moderate bandwidth tasks.
   - **Availability**: A reliable system should offer redundancy and adapt to evolving censorship tactics. Dynamic networks and backup options improve availability.
   - **Security**: The system must protect user privacy and resist interception. Encryption and protection against man-in-the-middle attacks are essential. Systems like `Stegozoa`, `Snowflake`, and `TorKameleon` prioritize strong security features.
   - **Resources**: The system should match the user's device capabilities and network conditions. Lightweight tools like `Snowflake` are efficient for low-resource environments.

   While `Snowflake` is currently the most accessible option for end users, it would be valuable to develop and deploy an accessible implementation of `TorKameleon` to provide more robust alternatives in the future.

2. **How does the usability of different censorship circumvention tools impact their adoption by non-technical users?**

   - `Snowflake` is the most accessible, requiring no user configuration and running seamlessly through browser extensions.
   - `Stegozoa` integrates as a library within WebRTC applications, simplifying use for those familiar with basic app installations.
   - `TorKameleon` has the potential to become a user-friendly option if its implementation is made more accessible.

Scenario-Based Security

1. **Under what circumstances might decentralized architectures like Snowflake or adaptable systems like TorKameleon provide better censorship resistance compared to more static solutions?**

   Decentralized architectures like `Snowflake` or adaptable systems like `TorKameleon` can offer better resistance to censorship in environments where censorship is dynamic and targeted. By distributing communication through constantly changing networks of nodes or proxies, these systems make it harder for censors to identify and block specific endpoints.

   Some specific scenarios where these systems excel include:

   - **When the censor targets known nodes**: In systems where entry nodes are publicly known, such as traditional Tor bridges, censors can easily block them. `Snowflake` mitigates this by using temporary proxies that frequently change, making it difficult for censors to block all available proxies.
   - **When facing sophisticated traffic analysis**: Static systems with predictable traffic patterns are vulnerable to detection. `TorKameleon` adapts its traffic patterns and uses encapsulation techniques, such as WebRTC-based covert channels or TLS tunnels, to evade traffic analysis.
   - **When censorship is intermittent or localized**: In environments where censorship varies by region or time, decentralized systems provide alternative routes for communication. If a proxy or node is blocked, `Snowflake` can seamlessly reroute traffic through other available proxies.
   - **When user resources are limited**: `Snowflake` is well-suited for users with limited resources, as it runs lightweight proxies within web browsers, eliminating the need for dedicated servers or complex configurations.

   These features make `Snowflake` and `TorKameleon` more resilient compared to static solutions, which are easier for censors to target and block.

2. **In what real-world scenarios could a censor exploit vulnerabilities in protocols that rely on steganography, such as Stegozoa, to disrupt covert communication?**

   In real-world scenarios, a censor can exploit the vulnerabilities of protocols that rely on steganography, such as `Stegozoa`, to disrupt covert communication in several ways:

   - **Media Manipulation**: A censor with control over WebRTC gateways can manipulate video streams through transcoding or applying image filters. These actions alter the DCT coefficients where `Stegozoa` embeds hidden data, leading to errors in message recovery on the receiving end. This manipulation can be subtle enough to avoid detection by regular users[10].
   - **Recoding Fragility**: Routine operations like video compression or re-encoding performed by WebRTC gateways can corrupt the steganographic data embedded by `Stegozoa`. This fragility can prevent the successful recovery of hidden messages.[10].
   - **Exploiting Protocol-Specific Weaknesses**: The algorithm used for embedding data, such as Syndrome-Trellis Coding (STC), is sensitive to changes in the embedding space. Even minor modifications to the least significant bits of the coefficients can result in cascading errors during message extraction.
   - **Embedding Space Fragility**: `Stegozoa` dynamically filters coefficients with values of zero and one during embedding. However, these coefficients may not remain static after recoding, preventing correct message recovery due to inconsistencies in the embedding space.

   The effectiveness of these censorship tactics depends on the sophistication of the censor, their available resources, and the specific architecture of the steganographic system. While `Stegozoa` continues to evolve to counter emerging threats, users must remain aware of potential vulnerabilities and take necessary precautions to protect their communications.

Ethical and Practical Considerations

1. **What challenges arise in balancing security and performance when implementing techniques like protocol obfuscation and steganography in real-world applications?**

   - **Security** often demands complex obfuscation, which can reduce **performance** (e.g., `Stegozoa`'s low throughput).
   - High-performance systems like `Protozoa` may sacrifice security by being vulnerable to traffic analysis or correlation attacks.
   - Striking a balance requires adaptive techniques that optimize both aspects based on user needs and network conditions.

2. **How can developers of censorship circumvention tools ensure adaptability to evolving censorship techniques and network conditions?**

   Developers can ensure adaptability by adopting a multifaceted approach that emphasizes flexibility, resilience, and continuous learning. Key strategies include:

   - **Modular Design**: Implement modular architectures for easy updates and integration of new techniques.
   - **Decentralization**: Use decentralized networks, such as those employed by `Snowflake`, to distribute communication and avoid single points of failure.
   - **Obfuscation Techniques**: Employ multiple methods of traffic obfuscation and dynamically switch between them based on network conditions.
   - **Steganography**: Integrate robust steganographic methods to hide communication within legitimate traffic, while addressing vulnerabilities to media manipulation and traffic analysis.
   - **Continuous Monitoring and Feedback**: Regularly monitor censorship tactics and gather user feedback to identify new challenges and refine tools accordingly.
   - **Collaboration**: Foster collaboration between developers, researchers, and users to share knowledge, identify vulnerabilities, and develop effective circumvention strategies.
   - **Ease of Use**: Prioritize user-friendly interfaces and minimal configuration requirements to ensure broad adoption, even by non-technical users.

   By incorporating these principles, developers can create adaptable and resilient tools that respond effectively to evolving censorship techniques and network conditions.

Comparative Effectiveness

1. **How do latency and throughput limitations in systems like TorKameleon influence their usability for real-time communication compared to more efficient systems like Snowflake?**

   - `TorKameleon`'s latency (530–655 ms) can hinder real-time communication.
   - `Snowflake` offers better latency and throughput due to its lightweight proxies, making it more suitable for real-time use.

2. **What practical steps can be taken to improve the resilience of systems vulnerable to media manipulation or transcoding, such as Stegozoa?**

- Use **error correction codes** to recover data after manipulation.
- Embed data in **keyframes** to withstand transcoding.
- Implement **static embedding spaces** known to both sender and receiver.

These steps and many others are discussed in much greater depth in the work of Cruzat La Rosa[10].

5 Future Trends and Challenges

The future of censorship circumvention tools will be shaped by the dynamic interplay of advancing technology, evolving censorship techniques, and the shifting needs of users. Below are key trends and challenges anticipated in this domain:

Advancing Censorship Techniques

Censors are likely to adopt more sophisticated technologies, including machine learning and big data analytics, to detect and block circumvention efforts. Techniques such as deep packet inspection (DPI) and advanced traffic analysis will present significant hurdles. Developers must innovate to counteract these measures, employing obfuscation, mimicry, and other advanced techniques to evade detection.

Artificial Intelligence (AI) as a Double-Edged Sword

AI will play a critical role in both enabling and combating censorship. While censors may utilize AI to identify patterns in network traffic, developers can leverage AI to:

- Enhance traffic obfuscation strategies by dynamically adapting to changes in network conditions.
- Detect and counteract censorship tactics in real time.
- Generate synthetic traffic and convincing decoys to confuse censors.

However, the arms race between AI-driven censorship and circumvention will require constant adaptation and resource investment.

Mobile-Centric Solutions

The proliferation of mobile devices as primary internet access points, particularly in heavily censored regions, necessitates the development of lightweight, resource-efficient tools optimized for mobile platforms. These tools must operate effectively on networks with low bandwidth and high latency.

Cloud Computing and Its Implications

Cloud-based infrastructure offers opportunities for scaling circumvention tools and hosting decentralized proxies. However, reliance on major cloud providers introduces vulnerabilities, as these providers may face pressure to restrict such activities. Decentralized and distributed cloud architectures may mitigate these risks.

Privacy and Security Concerns

As users become more aware of privacy risks, tools must prioritize:

- Strong encryption protocols to secure communications.
- Mechanisms to ensure user anonymity.
- Safeguards against data leaks and correlation attacks.

Balancing privacy with performance will remain a key challenge.

Improving Usability

To achieve widespread adoption, tools must be accessible to non-technical users. This involves simplifying configuration, offering intuitive interfaces, and minimizing the need for user intervention. Usability-focused design will enhance the impact of these tools, particularly in regions with low technological literacy.

Collaboration and Knowledge Sharing

The global community of researchers, developers, and advocates must collaborate to:

- Share insights on emerging censorship tactics and countermeasures.
- Develop interoperable standards for circumvention technologies.
- Promote open-source contributions to accelerate innovation.

Policy and Advocacy Efforts

In addition to technical innovation, advocating for policies that support a free and open internet is crucial. Efforts should focus on promoting digital rights, opposing restrictive legislation, and raising awareness of censorship issues globally.

Conclusion

The evolution of censorship circumvention tools will require a proactive and adaptive approach. By integrating cutting-edge technologies, prioritizing user needs, and fostering global collaboration, developers can address the challenges ahead and ensure unrestricted access to information for users worldwide.

6 Conclusion

This study analyses various technologies and methodologies designed to circumvent censorship in highly controlled networks, assessing their effectiveness, security and usability. Each system has unique strengths and weaknesses depending on censorship conditions and user capabilities. Tools such as Snowflake stand out for their accessibility and ease of use, while other systems such as TorKameleon and Stegozoa present innovative solutions but limitations in real-world scenarios.

The continuous evolution of censorship tactics, driven by technologies such as artificial intelligence, poses a constant challenge for the development of robust and adaptive tools. It is essential to combine technical advances with user-centric designs, prioritising security, performance and usability. In addition, collaboration between researchers, developers and digital rights advocates is crucial to address this dynamic landscape. This analysis not only provides a basis for future research, but also underlines the need for comprehensive strategies to ensure unrestricted access to information in an increasingly constrained global environment.

References

[1] Afonso Vilalonga . Torkameleon. `https://github.com/AfonsoVilalonga/TorKameleon`. Accessed: 2024-12-09.

[2] Diogo Barradas and Nuno Santos. Towards a scalable censorship-resistant overlay network based on webrtc covert channels. In *Proceedings of the 1st International Workshop on Distributed Infrastructure for Common Good*, page 37–42, 2020.

[3] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. Poking a hole in the wall: Efficient censorship-resistant internet communications by parasitizing on webrtc. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, page 35–48, 2020.

[4] BlizzardPlus. Code talker tunnel. `https://github.com/blizzardplus/Code-Talker-Tunnel/tree/master/source`. Accessed: 2024-12-08.

[5] Cecylia Bocovich Arlo Breault David Fifield and Serene Xiaokang Wang. Snowflake, a censorship circumvention system using temporary webrtc proxies.

[6] Gabriel Figueira, Diogo Barradas, and Nuno Santos. Stegozoa: Enhancing webrtc covert channels with video steganography for internet censorship circumvention. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, page 1154–1167, 2022.

[7] Bridger Hahn, Rishab Nithyanand, Phillipa Gill, and Rob Johnson. Games without frontiers: Investigating video games as a covert channel. In *2016 IEEE European Symposium on Security and Privacy (EuroSP)*, page 63–77. IEEE, 2016.

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

[9] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*, page 97–108, 2012.

[10] Adrian Cruzat La Rosa. No title. *Re-encoding Resistance: Towards Robust Covert Channels over WebRTC Video Streaming*, 2024.

[11] Afonso Vilalonga, João S. Resende, and Henrique Domingos. Torkameleon: Improving tor's censorship resistance with k-anonymization and media-based covert channels. In *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, page 1490–1495. IEEE, 2023.

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