Report: Onion Routing and Encrypted Communications

By Eshaan Prakash (0303204p)

COMP 3343 FA01 Data Comm & Comp Networks

Table of Contents

Introduction

This report explores the history of encrypted networks and onion routing with special emphasis on the TOR network, its purposes in present day society, and the technical workings behind such sophisticated network technology. The report also provides sample application examples showcasing the implementation of such concepts in 3 different python applications using various libraries relevant for TOR, network communication and system access to the networks. Additionally, the report compares Tor with other encrypted networks, such as I2P and Freenet, highlighting their differences in implementation and functionality

Key topics:

- History of encrypted networks and Onion Routing
- Tor network purposes and applications
- Comparison with other encrypted networks
- Implementations of such concepts and different use cases.

Onion Routing & Tor

History of Onion Routing and the Birth of TOR: Generation 2 Onion Routing.

A communications interface which has protection against eavesdropping and traffic analysis, Onion routing can be used for Internet application through proxies or by modifying the network protocol stack to connect to the network.

Onion routing initially started off in the mid 1990s in the United States Naval Research Laboratory (NRL) by researchers Paul Syverson, Michael Reed, and David Goldschlag as part of a project aimed at protecting classified government information and around 2001, the beta network began and the Edison Invention Award was presented for the invention.

2002 was when Generation 2 started developing and finished around October 2003, when the Tor network was deployed and Tor was released under the free and open MIT license.

Present day Tor.

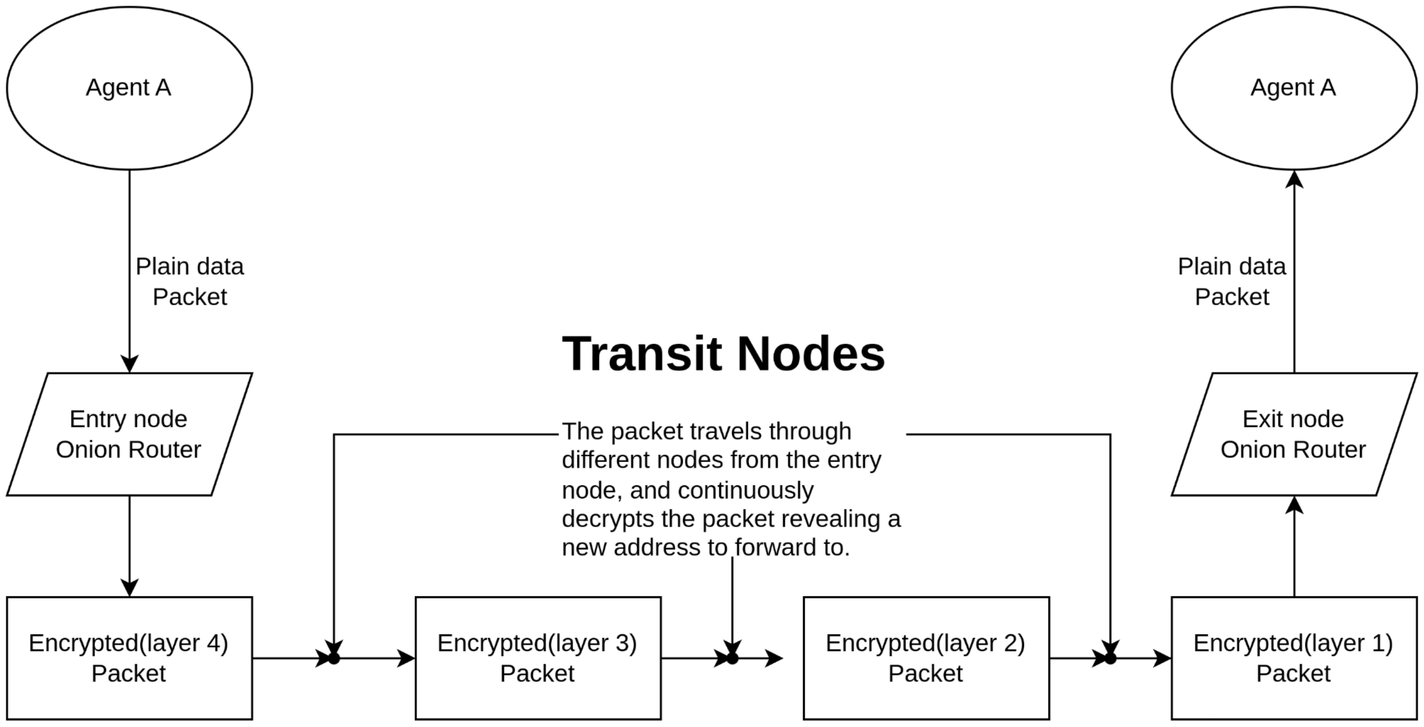
These days tor is used to bypass internet censorship and to resist mass government surveillance (NSA). Tor is also used to gain access to content which is generally unavailable in the public through means of piracy etc.

Behind the Scenes, Functionalities of tor.

Highly secure socket connections are made where the privacy of the connections are below the application layer. An application makes a connection to the onion routing proxy (entry node) which proceeds to make more connections with other onion routers. The packets which are being transferred from router to router only contain the encrypted data and the next node address, the succeeding node then decrypts a layer and gains another node address. Until the packet reaches the exit node and reaches the client in plain text format.

The data going through each transit node is different to maintain anonymity and if any connection is broken, all information about the connection is cleared.

There is another method of onion routing which tor does NOT implement in which there is a bump inserted into the TCP/IP Network Protocol stack so that all TCP traffic is routed


Agent A
Plain data Packet
Entry node Onion Router
Encrypted(layer 4) Packet
Transit Nodes
The packet travels through different nodes from the entry node, and continuously decrypts the packet revealing a new address to forward to.
Encrypted(layer 3) Packet
Encrypted(layer 2) Packet
Encrypted(layer 1) Packet
Exit node Onion Router
Plain data Packet
Agent A

through tor This is different from other proxies as normal proxy providers set up a server somewhere on the Internet and allow you to use it to relay your traffic but tor redirects the traffic through at least 3 nodes before reaching the destination and as there is a separate layer, no one can peek into the data.

Onion Services

Services which can be accessed over the tor network are called onion services and have the .onion suffix in their link instead of the usual .com, .net or .ca. As onion services are overlay networks over the TCP/IP so technically their 'IP address' is protected.

SOCKS

Socket Secure is a network protocol for communication between clients and servers through proxies. Operating at layer 5 of the OSI Model allows it to handle various network files.

The proxies are commonly used for:

- Enhancing privacy by masking the client's real IP address
- Bypassing geographical restrictions and censorship
- Improving security in remote access scenarios
- Web scraping, especially for dynamic content and APIs

Encryption Process of the Onion network

The process of encryption for the tor network is a complex procedure to maximise privacy and ensure anonymity with security.

The client first obtains a list of tor nodes to connect as entry, transit and exit nodes. It generates an ephemeral random symmetric key for each node using **AES-128-CTR.** The message is then encrypted multiple times using the keys of the respective nodes. The process is reversed when the destination server responds.

Alternative Encrypted Communications Network

I2P

The Invisible Internet Project, is a fully encrypted private network layer which is connected peer to peer, while hiding users from the server and server from the user. It uses encrypted unidirectional tunnels between you and your peers. It uses garlic routing, encrypting multiple messages together to promote privacy of data while using a distributed network database for peer discovery. It also has unidirectional tunnels instead of bidirectional circuits, doubling the number of nodes a peer has to compromise to get the same information.

Freenet

Freenet is a decentralized p2p network using a distributed network architecture to store and retrieve data, it is mainly used for data retention. Nodes are used to maintain the decentralized structure where each node is a computer participating in storing and sharing data. The data is stored in chunks through multiple nodes allowing anonymity similar to matrix. It also uses end to end encryption (AES-256) with RSA for key exchange and authentication, it also uses HMAC and the Diffie-Hellman key exchange.

Differences between Tor and..

Tor v I2P

- Tor uses a centralized directory system whereas I2P uses a network of nodes
- Tor uses a hierarchical routing topology while I2P utilizes the flat routing topology
- I2P is generally faster due to flat routing algorithms & decentralized nodes

Tor v Freenet

- Freenet also utilizes a network of nodes helping it setup a decentralized network.
- Freenet uses complex routing algos with specialized configs.
- Freenet is also primarily used for hosting and sharing content instead of the main use for anonymity like tor.

Implementation of Tor

Preliminary Setup

Pip Requirements

Be sure to install the requirements from the requirements.txt file using pip.

In some cases, a python venv is required.

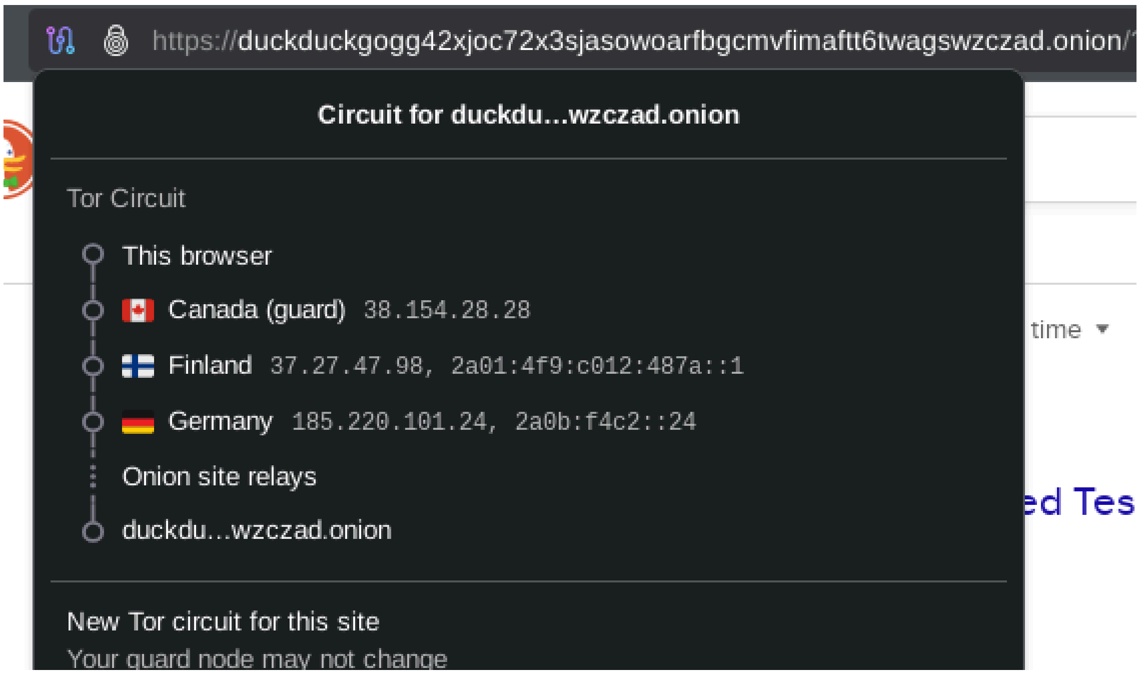
Torrc file

The torrc file located within the tor directory is the configuration used by tor to initialize the routing service. For the service to work, the torrc file must be modified by adding/uncommenting the following lines

```
Unset
SocksPort 9050 # Default: Bind to localhost:9050 for local connections.
DNSPort 5353
AutomapHostsOnResolve 1

ControlPort 9051
# test123 or test12
HashedControlPassword 16:4A489F4E38DFBB8660A440843C2D21DAA7CCB5628F8FF9E4EEA01B04EA

HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 127.0.0.1:5000
```

Port Scanning for Tor

The torPortScanner.py uses the scapy library to sniff and scan network packets on known ports used by tor to check for potential tor connections.
We will be using a simple Tor Browser set up on an Opensuse Tumbleweed Linux System with entry node set as:



https://duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzczad.onion/

Circuit for duckdu...wzczad.onion

Tor Circuit

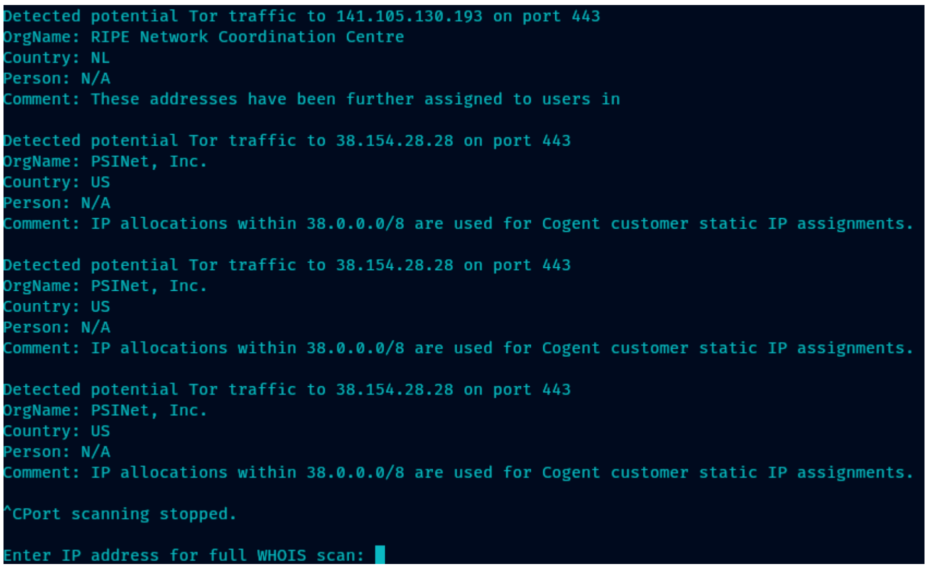
- This browser
- Canada (guard) 38.154.28.28
- Finland 37.27.47.98, 2a01:4f9:c012:487a::1
- Germany 185.220.101.24, 2a0b:f4c2::24
- Onion site relays
- duckdu...wzczad.onion

New Tor circuit for this site
Your guard node may not change

As you can see, the tor entry node (aka the guard node) is set up as 38.154.28.28

After running torPortScanner.py while this connection is active, you can see the output of the code to include the tor entry node only and not the transit nodes of Finland, Germany etc.

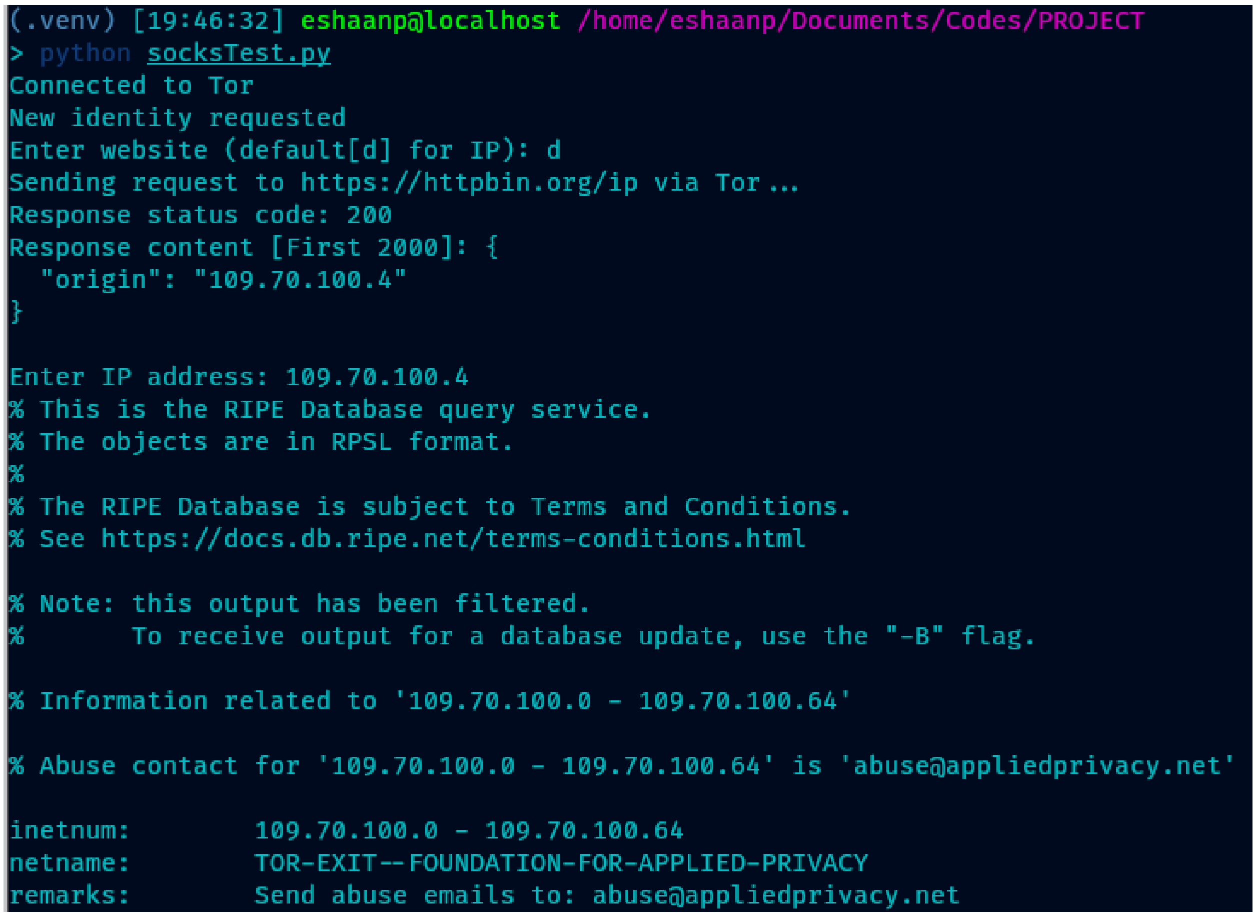
The code also provides the ability to find out the whois info for the IP of interest, which you can activate through a keyboard interrupt during the scan. Here in the results of the whois you can clearly see the tor node has been detected and information is provided of the host of the node. As the traffic itself is encrypted through TLS, the details of the packet are heavily modified compared to normal internet traffic. The WHOIS information also showcases how easy it is to obtain information of a domain service, which is highly improbable to duplicate on the onion network. This basically showcases how the destination IP address is hidden and not received (duckduckgo in this scenario).



```
Detected potential Tor traffic to 141.105.130.193 on port 443
OrgName: RIPE Network Coordination Centre
Country: NL
Person: N/A
Comment: These addresses have been further assigned to users in

Detected potential Tor traffic to 38.154.28.28 on port 443
OrgName: PSINet, Inc.
Country: US
Person: N/A
Comment: IP allocations within 38.0.0.0/8 are used for Cogent customer static IP assignments.

Detected potential Tor traffic to 38.154.28.28 on port 443
OrgName: PSINet, Inc.
Country: US
Person: N/A
Comment: IP allocations within 38.0.0.0/8 are used for Cogent customer static IP assignments.

Detected potential Tor traffic to 38.154.28.28 on port 443
OrgName: PSINet, Inc.
Country: US
Person: N/A
Comment: IP allocations within 38.0.0.0/8 are used for Cogent customer static IP assignments.

^CPort scanning stopped.

Enter IP address for full WHOIS scan:
```

Socks Test

The SOCKS implementation test also showcases how a simplified python application capable of connecting to tor can have a highly obfuscated IP address:



```
(.venv) [19:46:32] eshaanp@localhost /home/eshaanp/Documents/Codes/PROJECT
> python socksTest.py
Connected to Tor
New identity requested
Enter website (default[d] for IP): d
Sending request to https://httpbin.org/ip via Tor ...
Response status code: 200
Response content [First 2000]: {
  "origin": "109.70.100.4"
}

Enter IP address: 109.70.100.4
% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See https://docs.db.ripe.net/terms-conditions.html

% Note: this output has been filtered.
%       To receive output for a database update, use the "-B" flag.

% Information related to '109.70.100.0 - 109.70.100.64'

% Abuse contact for '109.70.100.0 - 109.70.100.64' is 'abuse@appliedprivacy.net'

inetnum:        109.70.100.0 - 109.70.100.64
netname:        TOR-EXIT--FOUNDATION-FOR-APPLIED-PRIVACY
remarks:        Send abuse emails to: abuse@appliedprivacy.net
```

The application creates a connection with the tor interface of the system, authenticates itself and sends a connection to a website of choice, in this case I chose https://httpbin.org/ip to showcase how the IP is different of the user agent (us) as the IP being shown is of the TOR exit node and not our IP. This showcases how secure the tor network actually is and how it can be used for multiple purposes ranging from security and administration to journalism and more.

Softwares like SecureDrop have been used as a whistleblower service providing secure communication between journalists and sources. It masks the user IP and location making it difficult to track the original sender.

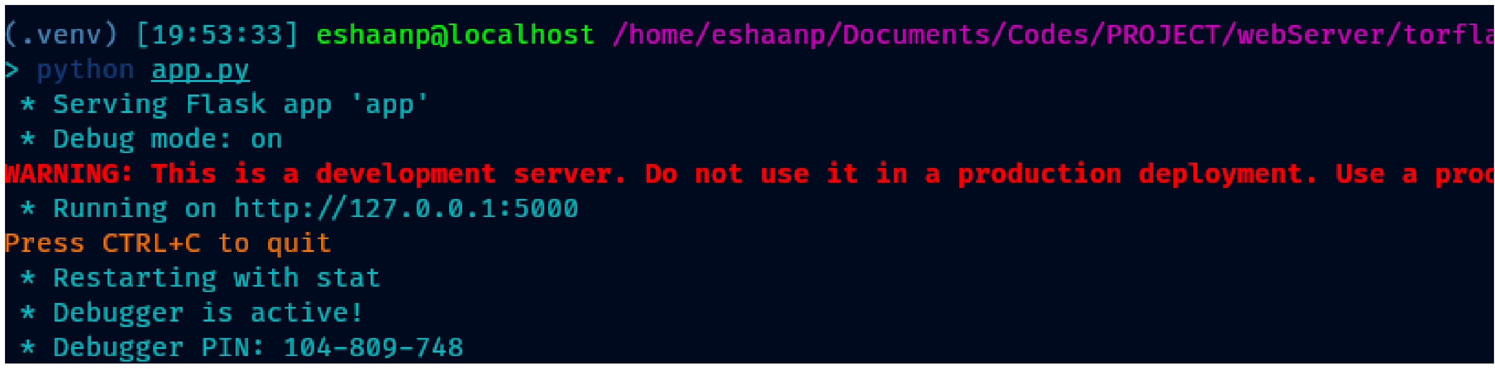
Onion service implementation.

TOR can also be used for simple purposes like hosting a web service which is easily accessible to other tor users.

app.py is a flask implementation of the web server with SQLAlchemy for database management with encryption functionalities, this is an example of an onion service which can be used for multiple purposes, in this case, the service is a message board service with user registration, with user linked message encryption. It uses a random 24 byte string as a secret key for session management, the Fernet Cipher suite is used to encrypt the chat communications data. It also provides a whois service as an example to showcase how whenever any tor user connects to the onion service, it cannot detect the IP of the user. The webserver works in tangent with the torrc file of the tor daemon.

You can obtain the name of your own web service using this command:

```
Unset
sudo cat /var/lib/tor/hidden_service/hostname
```



```
(.venv) [19:53:33] eshaanp@localhost /home/eshaanp/Documents/Codes/PROJECT/webServer/torfla
> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a proc
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 104-809-748
```

Advantages and Disadvantages

Benefits of using Tor:

- **Better Anonymity** - With efficient node traversal through flexible socket connections with multiplier proxies, tor manages to be one of the top contenders for maintaining anonymity in the digital world.
- **Encryption** - Using multiple levels of encryption tor manages to keep data secure whilst maintaining smooth transportation of said data. The so called ‘onion routing’ approach ensures that content is only available at the end node.
- **Decentralized** - Tor network is made up of thousands of volunteer operated servers allowing to circumvent censorship and maintaining protection against shutdowns and attacks.

Drawbacks of using Tor:

- **Slow speeds** - Due to the multiple server hops present in the routing, the data transfer speeds are comparatively slower than normal direct connections.
- **Entry-Exit Vulnerabilities -** The exit nodes are susceptible to traffic monitoring which can pose a risk for sensitive information, working together with entry node tracking, one could hypothetically fully trace a packet through decryption and tracing.
- **Not 100% foolproof** - The tor network is still vulnerable to timing attacks or browser vulnerabilities, as such events could compromise user privacy.

Conclusion

The onion router represents a great technological innovation towards the fields of cybersecurity promoting privacy and anonymity. Being the brainchild of the United States Naval Research Laboratory in the mid-1990s, it has evolved from a government-focused project to a critical tool for protecting digital privacy in an increasingly surveilled world.

Tor is capable of handling such tasks through its sophisticated onion routing mechanism, which provides multiple layers of encryption and anonymity. It could be and is utilized for various purposes like chat communication apps, encrypted data storage, etc.

Due to issues with digital surveillance throughout history, technologies like tor provide a fresh perspective on individual private technology and freedom. The future of such encrypted networks will involve continuous refinement of technology while keeping the values of maintaining privacy and allowing freedom to access the internet for sensitive occasions.

References:

- Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: The Second-Generation Onion Router: Defense Technical Information Center. https://doi.org/10.21236/ADA465464

- Collier, B. (2024). Tor: From the Dark Web to the Future of Privacy. The MIT Press. https://doi.org/10.7551/mitpress/14907.001.0001

- Tor Project | Set up Your Onion Service. (n.d.). Retrieved November 26, 2024, from https://community.torproject.org/onion-services/setup/

- Onion Routing: Executive Summary. (n.d.). Retrieved November 26, 2024, from https://www.onion-router.net/Summary.html#Problem

- Tan, Q., Wang, X., Shi, W., Tang, J., & Tian, Z. (2022). An Anonymity Vulnerability in Tor. IEEE/ACM Trans. Netw., 30(6), 2574–2587. https://doi.org/10.1109/TNET.2022.3174003

- Choorod, P., Bauer, T. J., & Aßmuth, A. (2024). Distinguishing Tor From Other Encrypted Network Traffic Through Character Analysis (No. arXiv:2405.09412). arXiv. https://doi.org/10.48550/arXiv.2405.09412