

# Modernizing the Tor Ecosystem

Alexander Færøy

July 24, 2022

MCH 2022



This presentation explains some of the **ongoing and upcoming efforts within The Tor Project to modernize our ecosystem.**

We will be **focusing on the Tor Network itself** and the components used to run the network.

We will not be covering efforts happening around Anti-censorship technology, Tor Browser, Localization, Metrics, UI, UX, and the other exciting things happening inside the Tor Project right now.

# About Me

- Core Developer at The Tor Project since early 2017. Team Lead of the Network Team since late 2019.
- Free Software developer since 2006.
- Co-organizing the annual Danish hacker festival **BornHack** on Funen.
- First hackercamp, out of many, was OHM in 2013.



# What is Tor?

- Online anonymity, and censorship circumvention.
  - Free software.
  - Open network.
- Community of researchers, developers, users, and relay operators.
- U.S. 501(c)(3) non-profit organization.



**Somewhere between 2,000,000 and 8,000,000 daily users.**

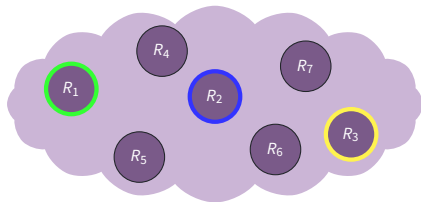


# How does Tor work?

Alice



The Tor Network



Bob



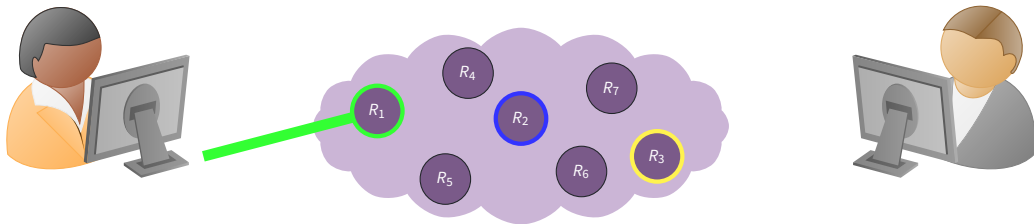
Alice picks a path through the network:  $R_1$ ,  $R_2$ , and  $R_3$  before finally reaching Bob.

# How does Tor work?

Alice

The Tor Network

Bob



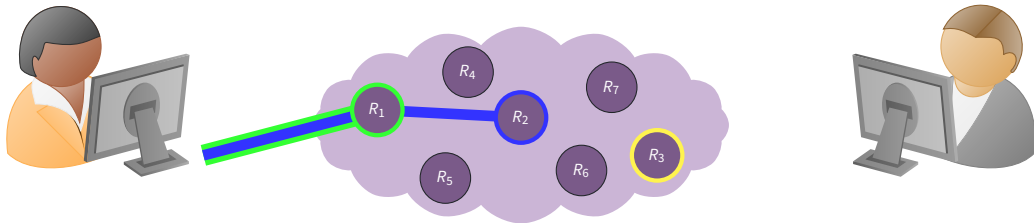
Alice makes a session key with  $R_1$ .

# How does Tor work?

Alice

Bob

The Tor Network



Alice asks  $R_1$  to extend to  $R_2$ .

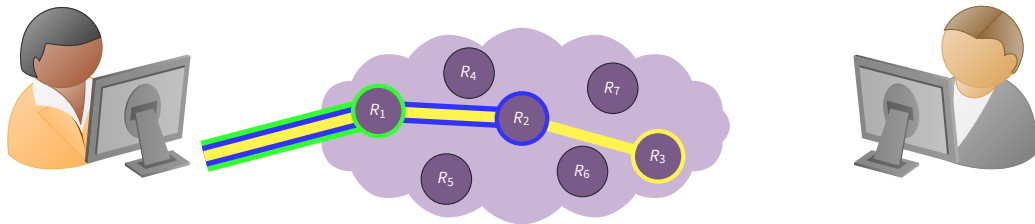


# How does Tor work?

Alice

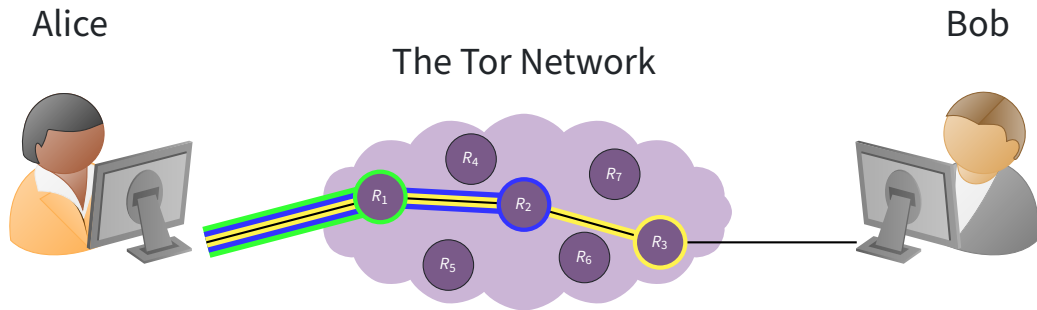
The Tor Network

Bob



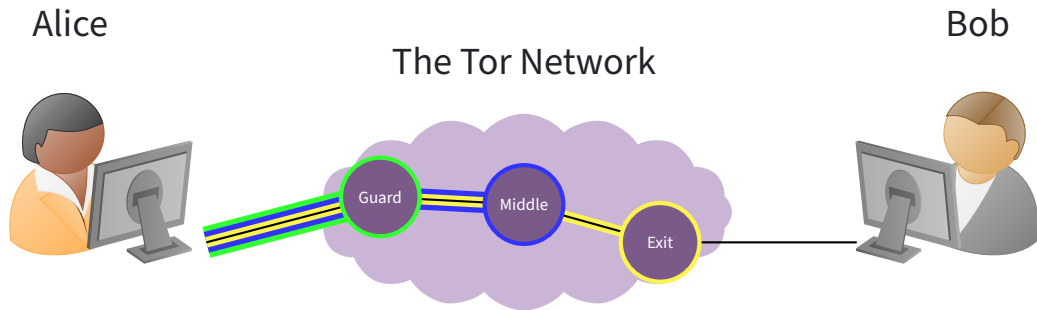
Alice asks  $R_2$  to extend to  $R_3$ .

# How does Tor work?



Alice finally asks  $R_3$  to connect to Bob.

# How does Tor work?



Speed > Blocking > Privacy > Security > UI

Speed > Blocking > Privacy > Security > UI

The performance of the Tor network: its throughput and latency.

Speed > **Blocking** > Privacy > Security > UI

Website blocking, captcha portals, denial of access, etc.

Speed > Blocking > Privacy > Security > UI

Privacy and anonymity guarantees by Tor.

Speed > Blocking > Privacy > **Security** > UI

Security of the different Tor components.



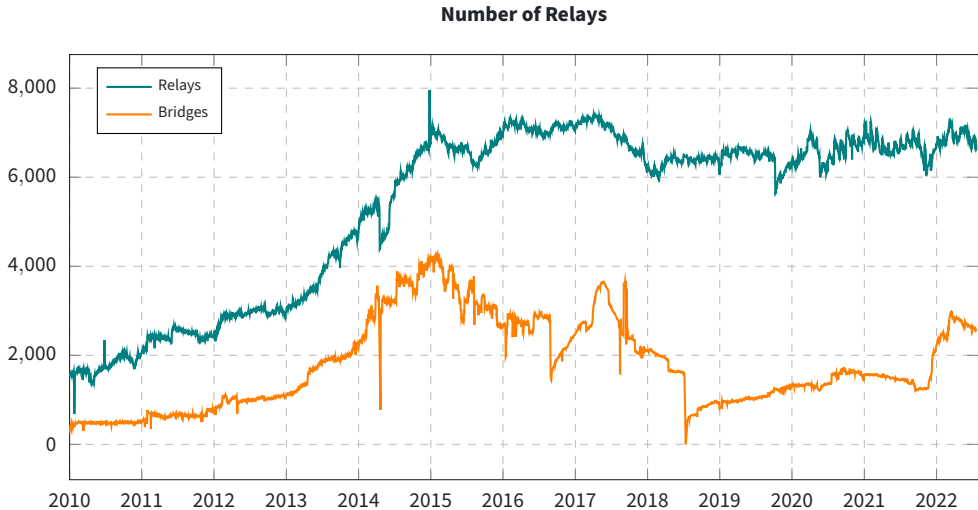
Speed > Blocking > Privacy > Security > UI

User interfaces of our products.

# The Tor Network

- An open network – **everybody** can join!
- Between 6000 and 7000 relay nodes.
- Kindly hosted by various individuals, companies, and non-profit organisations.
- 9 Directory Authority nodes and 1 Bridge Authority node.

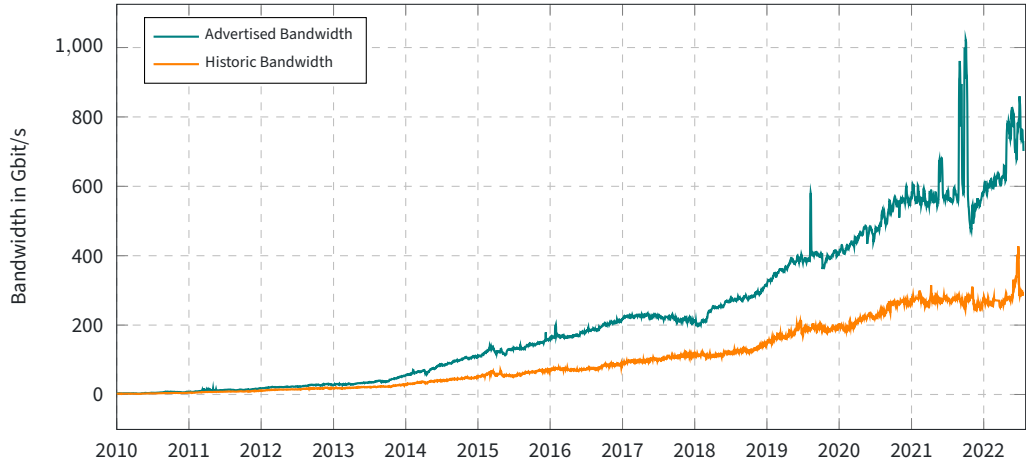
# The Tor Network



Source: [metrics.torproject.org](https://metrics.torproject.org)

# The Tor Network

**Total Relay Bandwidth**



Source: [metrics.torproject.org](https://metrics.torproject.org)

# Packed and Fragmented Cells

**Packed cells** will allow us to consolidate multiple smaller cells into fewer cells.

**Fragmented cells** will allow us to split payloads larger than 498-bytes into multiple cells.

See [Proposal #340](#).

# Post-quantum Cryptography

We need to apply post-quantum cryptography to two layers of the Tor protocol:

**The TLS layer:** Protects against an adversary that is able to intercept relay communication.

**The Tor Circuit layer:** Protects against an adversarial relay intercepting communication.

Fragmented cells are needed due to the large payloads used by the available cryptographic post-quantum handshakes.

# Congestion Control

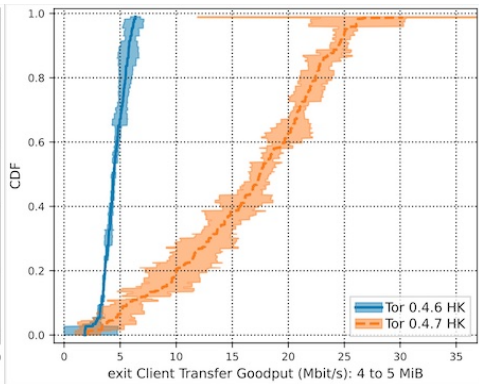
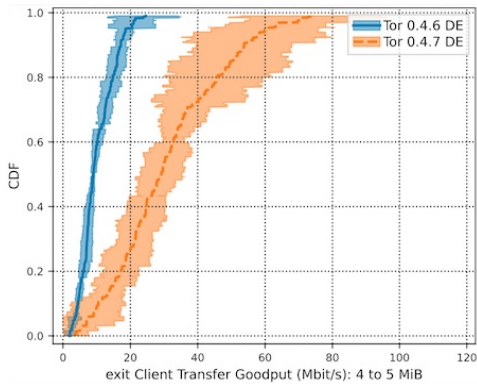
We implemented three congestion control algorithms: Tor-Westwood, Tor-Vegas, and Tor-NOLA. All of them are available in **Tor 0.4.7**.

Both Tor-Westwood and Tor-NOLA exhibited ack compression, which caused them to wildly overestimate the Bandwidth-Delay Product, which lead to runaway congestion conditions.

Google's BBR algorithm also suffers from these problems, and was not implemented in Tor.

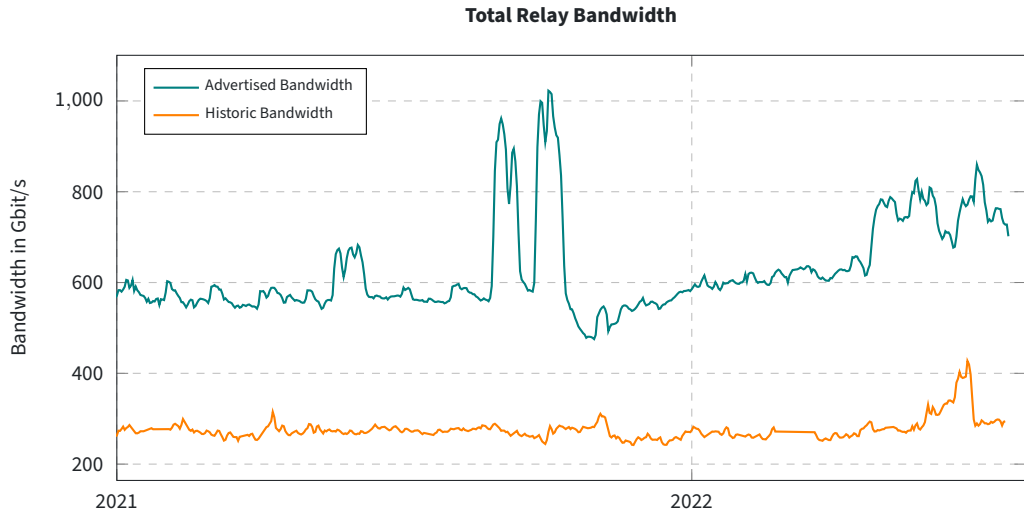
# Congestion Control

**Tor-Vegas** performed beautifully, almost exactly as the theory predicted, as seen in the results from **Shadow**.

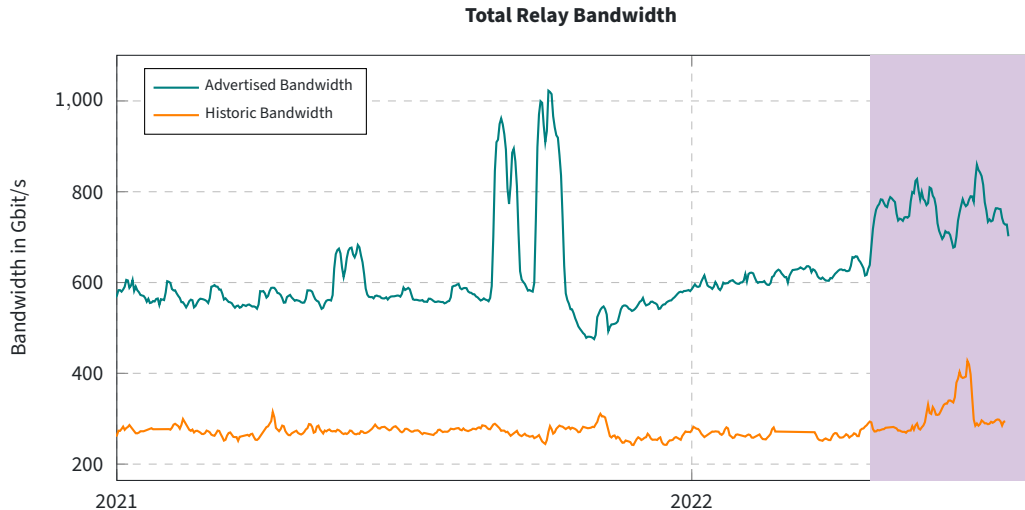




# Congestion Control



# Congestion Control



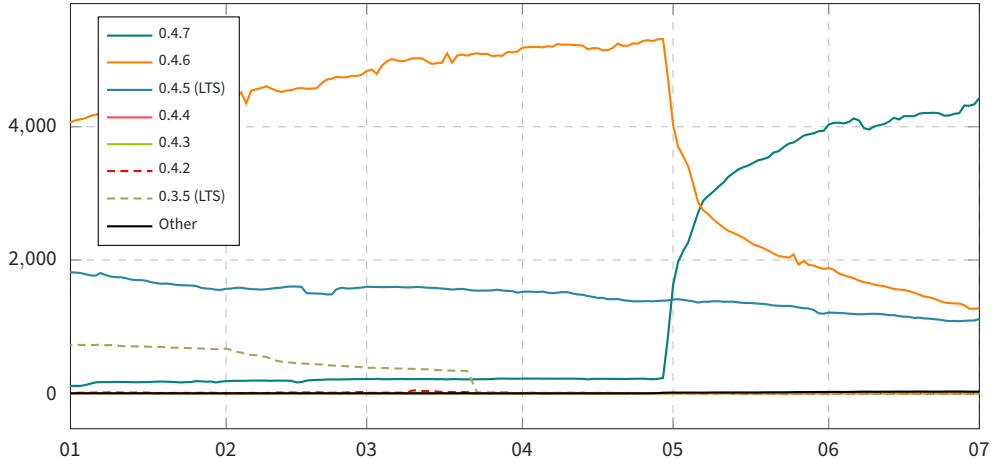
# Ongoing Denial of Service

The ongoing Denial of Service against the Tor network in the last couple of weeks have made it drastically harder to analyse the impact and tuning opportunities related to the deployment of congestion control in the network.

Ongoing efforts to reduce the impact of Denial of Service attacks is helping, but it continues to be a bit of a Whac-A-Mole game.

# The Tor Network

Relay Versions Seen During 2022



A massive **thank you** for upgrading to **Tor 0.4.7** so quickly!



# Further Tuning

We will continue tuning the Congestion Control subsystem in Tor in the near future.

One planned project is changing the bandwidth cut-off values for the **Fast** and **Guard** flags for relays.

# Congestion Control

Onion Service operators will also benefit from upgrading to **Tor 0.4.7**.

Relay Operators: be prepared to Set Bandwidth Limits

For more details, please read Mike Perry's blog post on Congestion Control at [blog.torproject.org/congestion-contrl-047](https://blog.torproject.org/congestion-contrl-047)

The goal is to overcome some of Tor's network performance bottlenecks using **traffic splitting**.

Current work specified by David Goulet and Mike Perry in Tor's **Proposal #329**.

Based on work by Masha'al AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg in the paper **The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting**.

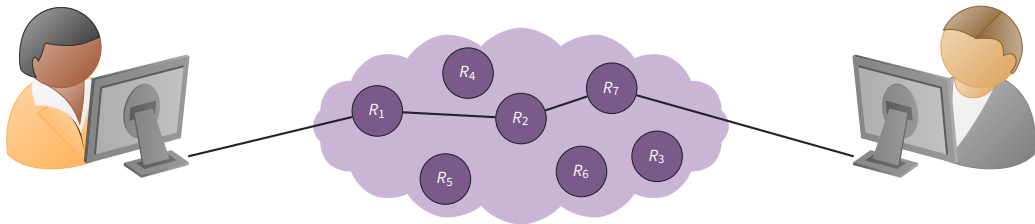


# Conflux

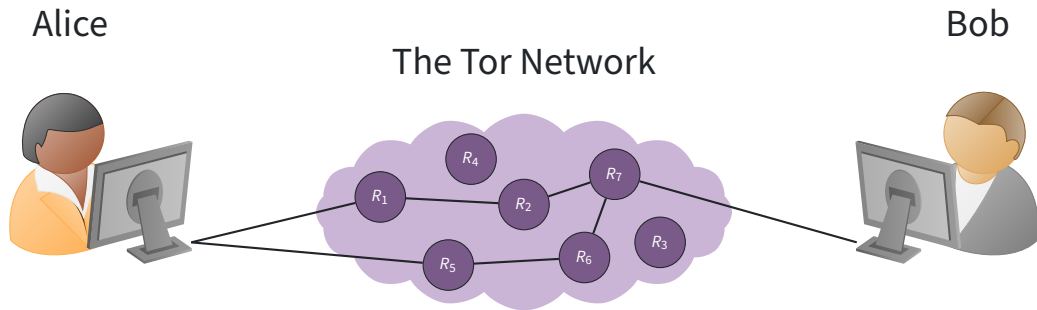
Alice

Bob

The Tor Network



# Conflux



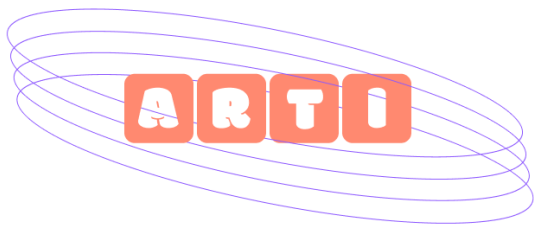
# Proof of Work for Onion Services

Implement PoW for Onion Services that can dynamically enable, disable, and adjust the difficulty of the system if pathological situations appears.

Make the cost of attacking an Onion Service higher.

A big thanks to **tevador** for all the help here!

See [Proposal #327](#).



Arti is our new Tor implementation in the Rust programming language.

Focus on building a library to work with the entire Tor ecosystem:

- Embed the Arti client into your own application.
- Parsing different Tor related network objects.
- Onion Services.

But, why rewrite Tor?

Writing "safe C" is costly, and prone to mistakes:

21 out of 34 of Tor's TROVE's was related to memory issues allowed by the C programming language.

Most of the Network Team at Tor is very excited about Rust, and was interested in spending more time writing software in it.

# Arti Roadmap

<b>0.1.0</b>	API stability.
<b>1.0.0</b>	Usability, performance, and stability.
<b>1.1.0</b>	Anti-censorship.
<b>1.2.0</b>	Onion services.
<b>2.0.0</b>	Ready to replace the C client.
<b>Future</b>	Relay, bridge, directory authority, etc.

# Arti and Legacy Tor

Currently, **3 out of 7 members of the Network Team are working full-time on Rust and Arti deliverables.** We aim to have the entire team work in this space as soon as possible.

We will **reduce feature additions in C Tor** drastically and will not be adding more Long-Term Support Tor releases.

We will **continue to support C Tor** until Arti can replace the currently used C Tor implementation.



# UDP Support in the Tor Network

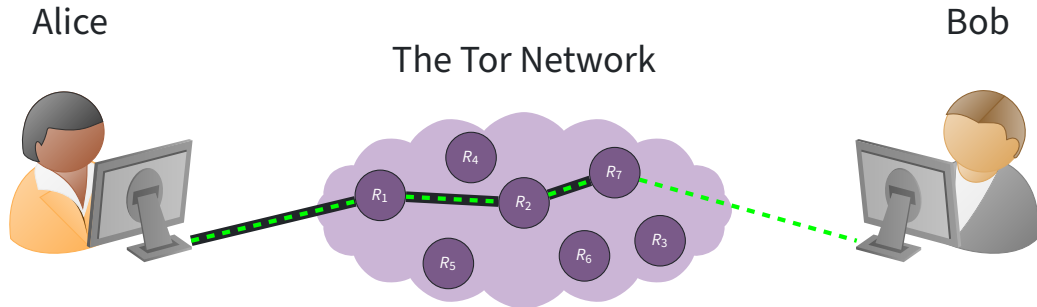
Support UDP for Tor clients and Exit nodes to allow support for modern internet applications such as: Crypto wallets, streaming, VoIP, and *hopefully* WebRTC-based applications.

Client and Exit nodes will require upgrades for deployment.

Use Tor's Congestion Control system to decide when to drop packets at the edges.

Specification work being tracked in [torspec#73](#) on [Tor's Gitlab](#).

# UDP Support in the Tor Network



# VPN using Tor

Allow users to use Tor with applications that are not necessarily Tor-aware.

The initial focus will be on the **Android** platform, but more platforms should become supported over time.

Our current **goal is to release the application in 2023.**

Build a layer 3 packet engine library in Rust for handling:

- Read and write IP packets from TUN devices.
- Multiplexing between TCP/UDP flows and Arti's TCP/UDP socket interface.
- Onion Service connectivity using cookie responses from DNS.
- Basic filtering mechanism for disallowing certain flows.

We need to expand the usual IP 5-tuple with additional flow metadata for isolation purposes:

- Application UUID (Unix User ID on Android) and its name.
- Hostname (if any).
- DNS cookie (for looking up cookie IP to Onion Service identifier).

# How can you help?

- Run a Tor relay or a bridge!
- Teach others about Tor and privacy in general.
- Find, and maybe fix, bugs in Tor software.
- Test Tor on your platform of choice.
- Work on some of the many open research projects.
- Donate at [donate.torproject.org](https://donate.torproject.org)



# Relay Operators Meetup

Let's meet here at MCH 2022!

**Tonight** at **21:00** in the **c-base village**.



[wiki.mch2022.org/Session:Tor\\_Relay\\_Operator\\_Meetup](https://wiki.mch2022.org/Session:Tor_Relay_Operator_Meetup)

# Questions?

✉ ahf@torproject.org

📧 @ahf@mastodon.social

🐦 @ahfaeroey

🔑 OpenPGP:  
1C1B C007 A9F6 07AA 8152  
C040 BEA7 B180 B149 1921





This work is licensed under a  
Creative Commons  
Attribution-ShareAlike 4.0 International License

