Engineer

Tor

a network that routes traffic through volunteer relays

What it is

Tor keeps the question of who is talking to whom away from anyone watching the wire, by passing each connection through a chain of machines so that none of them learns both where the traffic started and where it ends. The chain is chosen by the client, a program called tor that runs on the reader’s own computer, and it is usually three relays long: a guard that the client keeps reusing as its way in, a middle relay, and an exit that makes the final connection to the destination. The client builds this circuit one hop at a time and agrees a separate key with each relay, then wraps every piece of data in one layer of encryption per hop. Each relay removes its own layer and learns only the neighbour before it and the neighbour after it. Those nested layers are the onion in the title of the design paper, The Second-Generation Onion Router. In conversation the relays as a whole are the Tor network, tor in lower case is the program, and Tor Browser is the browser packaged to use it.

Data moves in cells of a fixed size, so a relay sees a stream of uniform blocks rather than the shape of the original packets. Several TCP streams can share a circuit when their isolation requirements are compatible. The client builds circuits both in advance and on demand, and retires them according to age and use; a long-lived stream can remain on its original circuit, and some isolation settings allow longer reuse. Applications reach the client through a local SOCKS proxy, and only TCP travels this way; the design paper states plainly that it does not tunnel protocols such as UDP. Volunteers run the relays, each publishing a descriptor with its keys, address, bandwidth and exit policy; a small set of directory authorities votes on which to list and signs the result as a consensus document that every client fetches before picking a path.

Use the concept in a decision

Put “What is Tor?” to work in a real decision. We can help relate the concept to the system you are building or buying. Wi-Fi, firewalls and remote access set up so staff can work from anywhere and only the right people get in.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Managed 30 Ubuntu Linux VPS instances — 30 Linux servers managed, with recovery planned for

Managed 30 Ubuntu Linux VPS instances Get in touch →

Why it matters

For anyone who runs a public website, Tor is first a fact about the audience. A reader who arrives over Tor appears to come from an exit relay’s address rather than their own, and the list of relays is public by design, because every client needs it to choose a path. A site can therefore recognise Tor traffic and treat it differently. The Tor Project’s own position is that operators have the right to block it, even if it hopes they will not. The decision that remains is what to do with those readers: serve them like everyone else, slow them down, put a challenge in front of them, or give them a way in that never leaves the Tor network at all, which is what an onion address does.

It pays to be exact about what Tor promises, because its design paper lists what it leaves out. It does not claim to defeat an observer who can watch both ends of a connection and compare timing and volume, and the project’s documentation adds that researchers know no practical low-latency design that reliably does. The basic protocol does not conceal Tor use itself; bridges with pluggable transports add ways to disguise that traffic. And Tor does not encrypt the last stretch, from the exit relay to the destination: the application needs encryption of its own for that hop.

Set beside a VPN, the trust is arranged differently. A VPN provider sees both the user and the destination and is asked to look away. On Tor no single relay sees both. For most paths, the selection rules keep relays from the same declared family apart and avoid a shared IPv4 /16 or IPv6 /32 subnet by default; there are documented exceptions and configurable subnet limits. The price is latency, since an ordinary exit circuit crosses three volunteer machines, so pages that are light and need few round trips serve Tor readers best. Offering an onion address needs a tor process and a private key to guard, but no open inbound port, which is why it fits on small infrastructure.

What it looks like in practice

The company's own site can be read over Tor. The command-line page publishes the onion address next to a `curl` command that fetches the Markdown text through a local tor client. The address is deliberately plain `http://` : a version 3 onion address encodes the service's identity public key together with a checksum and version byte. Tor authenticates the service and encrypts the circuit before HTTP is spoken. A public web certificate is not needed for that onion-address authentication.

An onion service needed room for a tor process on the company's September 2026 public-services host: a single core with about 464 MB of usable memory that already terminated TLS for seven domains and ran a git forge. Room for that process was measured rather than argued, over a recorded day of samples on that 512 MB host: the smallest amount of available memory seen was about 310 MB of the 464, swap use was 2.6 per cent and the processor sat idle 99.7 per cent of the time, so Tor went on beside the rest.

The privacy notice records the company's choice about counting readers. The Gemini and Gopher applications keep no access log, and their daily counters record only aggregate request totals. Gopher also has an operating-system connection log containing the source address; that log expires after 30 days. Over Tor the company says it counts nothing at all and offers no setting to enable it. That is an operator's privacy choice: Tor hides a reader's network address, but does not prevent a service from counting requests. The notice says the company would rather not know than be able to.

Where it goes wrong

The name is looked up outside Tor. A program that sends its connections to a SOCKS proxy may still resolve the hostname itself first, through the ordinary resolver, and that lookup tells the local network and the DNS provider which site is being visited. The connection then travels through Tor, but the destination has already leaked. In `curl` the difference is a single option: with `--socks5` the name is resolved locally, with `--socks5-hostname` it is handed to the proxy, and the exit relay resolves it.

```
# Resolves the name locally, then connects through the proxy
curl -s --socks5 127.0.0.1:9050 https://example.org/
# Hands the name to tor; the exit relay looks it up
```

```
curl -s --socks5-hostname 127.0.0.1:9050 https://example.org/
```

Plain text after the exit. Tor encrypts the hops between relays, not the last one. The exit opens an ordinary TCP connection to the destination and sends what the application gave it, so a login form submitted over plain HTTP is readable by that exit's operator and by anyone on the network beyond it. Serve HTTPS for everything that leaves the Tor network. An onion service sidesteps the question because its traffic never leaves.

One address stands for many readers. Rate limits and bans keyed on the client address treat an exit relay as a single visitor, though many people use it at once. A limit tuned for one person locks out everyone whose circuit happens to end there, and a ban earned by one abuser lands on all of them. Decide on purpose how exit addresses are treated instead of letting a generic per-address rule decide.

Identity travels inside the circuit. Tor hides the network address and does nothing about what the application says. Signing in to a personal account, a distinctive browser setup or a document that fetches something on opening links the session to a person, however many relays sit in between. The project's own documentation notes that application traffic still needs scrubbing on the user's side, which is the job Tor Browser takes on and an ordinary browser pointed at the proxy does not.

An exit relay on a home connection. Guard and middle relays rarely draw abuse complaints. The exit is where destinations see traffic coming from, so complaints and the occasional legal enquiry go to whoever runs it, and the Tor Project's relay guide says not to run one from home. A volunteer with a single home address can run a bridge instead, an unlisted relay that the same guide recommends for that situation.

Expecting UDP to work. Voice calls, many games and tools that send their own DNS lookups rely on UDP, and Tor carries only TCP streams. Those programs do not work through it, and one that is not forced through the proxy may simply connect directly, outside Tor, without saying so.

Further reading

- Tor Specifications — the protocol, path selection, directory and onion service specifications in one place.
- Tor: The Second-Generation Onion Router — the design paper: circuits, cells, exit policies, and an explicit list of what the design does not attempt.
- Tor Path Specification — how a client chooses its guard, middle and exit, and which relays may not share a path.
- Types of relays on the Tor network — guard, middle, exit and bridge relays, with the legal exposure each one carries for its operator.
- HTTPS encryption and Tor — an interactive view of what an eavesdropper sees with Tor, with HTTPS, with both and with neither.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is Tor?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/tor/