Engineer

Onion service

an address that works only through Tor

What it is

An onion service is a server that readers reach inside the Tor network by a name built from the server's own public key, instead of a DNS name that points at an IP address. People often shorten it to an onion, one tor daemon can run several onion services side by side, and the specification still uses the older name, hidden service.

The name is 56 characters of base32 followed by `.onion`. Decoded, it is 35 bytes: the service's 32-byte ed25519 identity key, a two-byte checksum taken from a SHA3-256 hash of that key, and a version byte set to 3. Because that byte comes last, every current address ends in the letter d. Nobody chooses the label; it is the key written out. The version 2 names before it were 16 characters cut from a SHA-1 hash of an RSA-1024 key, and tor dropped them in release 0.4.6.1-alpha, in 2021.

Reaching the service takes two meeting places. The service picks a few relays as introduction points, keeps circuits open to them, and uploads a signed descriptor listing them to directory relays. The descriptor is filed under a blinded key that changes each period and its body is encrypted, so a relay storing it without knowing the address cannot read the list. A client that holds the address fetches the descriptor, checks the signature against the key in the name, picks a relay as rendezvous point, and asks the service through an introduction point to meet there. The service builds its own circuit to that relay and completes the handshake, and the relay joins the two circuits. The path crosses six relays, three chosen by each side, and neither end learns the other's IP address. An optional mode, now called restricted discovery and formerly client authorization, also encrypts the descriptor for each authorised client's key, so only those clients learn where the introduction points are.

Use the concept in a decision

Put “What is an onion service?” 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

The name does the job that DNS and a certificate authority share for an ordinary site, and needs neither. Verifying the descriptor, and completing the handshake, tells the client that it is talking to whoever holds the private key behind the address. RFC 7686, which reserves `.onion` as a special-use domain name, describes such names as combining location and authentication, and draws the consequence: an onion name cannot be registered, assigned, transferred or revoked. Owning one means controlling the key pair, nothing more.

Encryption comes from the same design. Traffic to an onion service never leaves the Tor network, so there is no exit relay at the far end and no last stretch in plain text. Plain `http://` on an onion therefore carries a page encrypted end to end, to a host whose identity the client checked against the name it typed.

The service only ever dials out, to its introduction points and to each rendezvous point, so it needs no open inbound port and works behind NAT. Adding an onion leaves the host’s list of open ports as it was.

Readers still have to learn the address. A site can announce it with an `Onion-Location` response header, and Tor Browser then shows a purple pill in the address bar leading to the onion copy of the page. The Tor Project sets three conditions: the value is an `http:` or `https:` URL with an onion host, the page sending it arrived over HTTPS, and that page is not itself on an onion. A meta tag can carry the same announcement where headers cannot be set, but without the path, so every page points at one fixed URL.

What it looks like in practice

Publishing the company's onion mirror under a readable address is written up step by step. The daemon comes from the distribution's own package, with its outbound proxy port switched off. The private key behind the address was generated off the host and has never reached the repository or a laptop; the role declines to overwrite an identity it did not generate, so a key placed by hand is used as it is. The address is a declared value in the inventory, and every converge asserts that the identity file on the host matches it, so a swapped key with a stale declaration stops the run. Both secret guards catch the key file, after a measurement showed that the obvious key-file pattern matched a deploy key and missed this one.

A readable prefix can be found but never chosen. Each extra character multiplies the expected search by 32, so the eight letters of `engineer` take about 1.1 trillion keypairs on average. Retiring the random address the mirror first shipped with was a staged cutover, so the published address kept answering throughout. One step looked final and was not: the key landed, the converge reported no change, and the new address stayed unpublished because nothing named its directory. The role now fails on any identity on disk that no service names.

The credits page prints the full address, `engineer2ezpeji52ipphljcflzyhetvy5r6tnqubtq45nzlxxv6rsqd.onion`, beside the Gemini and Gopher ones. The four front doors share one folder of files and one deploy, so none of them can drift from the others, and the ordinary site sends `Onion-Location` so Tor Browser offers the onion unprompted. What the reader's own network sees depends on the door: over Gopher, which page is read; over Gemini, which address was asked for; over the onion, neither. The readable prefix is a convenience rather than a proof, so the whole address is the part to compare.

The address's structure can be checked offline. These lines decode it and recompute the checksum, and both comparisons print `True`:

```
import base64
import hashlib

label = "engineer2ezpeji52ipphljcflzyhetvy5r6tnqubtq45nzlxxv6rsqd"
raw = base64.b32decode(label.upper())
```

```
key, checksum, version = raw[:32], raw[32:34], raw[34:]
digest = hashlib.sha3_256(b".onion checksum" + key + version).digest()
print(version == b"\x03", digest[:2] == checksum)
```

A valid checksum shows that the address is well formed, not whose it is. That part comes from reading it on the company's own HTTPS pages.

Where it goes wrong

The key is the only deed. With no registrar behind the name, a lost private key means a lost address, and a copied one lets someone else answer at it. The Tor Project's setup guide says to keep the key files private and to back the key up if the service is meant to last. Treat it as a signing key, not configuration.

Reading only the first letters. RFC 7686 warns that an attacker can find keys whose names look like the intended one, and that the risk is larger because the names mean nothing to a person. It counts readable names, bought with considerable computing, as one remedy, but a prefix of eight characters leaves the other 48 as opaque as before. Compare the whole address, or arrive through a bookmark or a link on the site's own HTTPS pages.

The server names itself. Tor hides where the service runs, but the software behind it can still say. RFC 7686 lists a server leaking its identity in an application-level message among the ways the protection fails, and an error page printing the machine's real hostname is one. The setup guide also recommends passing traffic to the web server over a Unix socket rather than TCP, so the service does not leak onto the local network:

```
HiddenServiceDir /var/lib/tor/my-website/
HiddenServicePort 80 unix:/var/run/tor/my-website.sock
```

It matters most where the location itself is the secret.

The name goes to DNS. Software that does not speak Tor asks the ordinary resolver about an onion name, which tells the local network and the DNS provider which onion was wanted. RFC 7686 tells such software to fail without the lookup, and DNS servers to answer that the name does not exist. Hand the name to the Tor client instead.

An `Onion-Location` pill that never shows. A header set only on the plain-HTTP redirect breaks the HTTPS condition, and one with a mistyped address sends readers nowhere. Check the header on the final HTTPS response.

Further reading

- Tor Rendezvous Specification, version 3 — the whole protocol: identity and blinded keys, descriptors, introduction and rendezvous.
- Encoding onion addresses — the formula that turns a public key into a 56-character name, with example addresses.
- How do Onion Services work? — a walk through introduction points, descriptors and the rendezvous, step by step.
- Set up Your Onion Service — the torrc lines, the files tor creates, and advice on keys and on location leaks.
- Onion-Location — header and meta-tag setups for Apache, Nginx and Caddy, and the conditions the header must meet.
- RFC 7686 — why `.onion` is reserved, and what resolvers, DNS operators and registrars must do with it.

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 an onion service?” 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/onion/