T2 Privacy in Communications: Onion Routing

104369: Privacitat de Dades i Seguretat

G. Navarro-Arribas
UAB -- DEIC
v.20261020

©G. Navarro-Arribas – (2026)

Onion Routing

Tor
- Tor Relays and circuits
- Security properties and attacks
- Comments and future directions

©G. Navarro-Arribas – (2026)

Onion Routing

Tor
- Tor Relays and circuits
- Security properties and attacks
- Comments and future directions

©G. Navarro-Arribas – (2026)

Onion Routing: Core Mechanism [2]

- Client selects a path of L relays: $(R_1, \ldots, R_L)$.
- Establishes a **distinct symmetric key** K_i with each R_i.
- Constructs a **layered ciphertext**:

$$C = E_{K_1}\big(E_{K_2}(\cdots E_{K_L}(M)\cdots)\big)$$

- Each relay peels *one* layer and learns only the *next hop*.
- Result: No single relay can link *sender* and *receiver*.

Key Establishment: Original Onion Routing

- Client obtains each relay's **long-term public key** PK_i from a directory.
- Client generates a fresh symmetric key K_i for each relay and delivers it by **asymmetric encryption**:

$$\text{onion} = E_{PK_1}\big(K_1,\ R_2,\ E_{PK_2}(K_2,\ R_3,\ \ldots)\big)$$

- Each relay decrypts its layer with its private key, recovers K_i and the next hop, then forwards the inner onion.
- **Simple, non-interactive**: the whole structure is pre-computed by the client before sending anything.
- **Limitation**: uses long-term keys $\Rightarrow$ no *forward secrecy* (compromise of SK_i later reveals past K_i).

Key Establishment: Tor

- For each hop i, client and R_i derive a fresh session key K_i.
- Typically via **ephemeral Diffie–Hellman** (ensures *forward secrecy*).[1]
- Keys are independent; circuit is built **telescopically**: each new hop is added *through* the already-established partial circuit.

[1] Tor uses ephemeral X25519 (Curve25519) key pairs via the *ntor* handshake (since Tor 0.2.4).

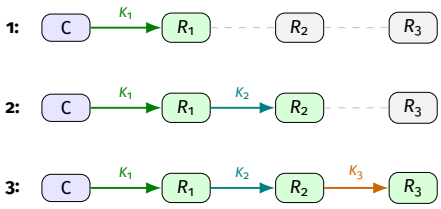
Circuit Construction: Telescoping

1. **Hop 1 (direct):** client $\xrightarrow{\texttt{CREATE}(pk_1)}$ R_1; R_1 replies directly. Client derives K_1.
2. **Hop 2 (via R_1):** client sends $E_{K_1}[\texttt{EXTEND}(R_2,\ pk_2)]$ to R_1.
   - R_1 peels its layer, sees EXTEND for R_2, forwards.
   - R_2 replies; R_1 relays back. Client derives K_2.
   - R_1 sees pk_2 and R_2's reply pr_2, but cannot derive $K_2 = \mathrm{KDF}(g^{sk_2 \cdot sr_2})$: lacks both sk_2 and sr_2.
3. **Hop 3 (via R_1, R_2):** client sends $E_{K_1}(E_{K_2}[\texttt{EXTEND}(R_3,\ pk_3)])$ to R_1.
   - R_1 peels K_1, passes to R_2; R_2 peels K_2, sees EXTEND for R_3, forwards.
   - Reply propagated back; client derives K_3.

Privacy invariant (CDH)

R_i sees both DH public values for hop $i{+}1$ but cannot derive K_{i+1}: it holds neither sk_{i+1} nor sr_{i+1}. EXTEND messages for hops $i{+}2, \ldots, L$ are encrypted under keys R_i lacks.

Telescoping: Circuit State After Each Step



Reading the diagram

Solid arrow = session key established between **client** and that relay; dashed = not yet in circuit.
In the data phase the C–R_1 link carries all L layers; each hop peels one.
K_i is known *only* to the client and R_i (not to $R_{i\pm1}$).

Correctness and Security Intuition

- **Correctness:** successive peeling reveals the intended route and payload.
- **Confidentiality:** without K_i, layer i is IND-CPA secure[2].
- **Integrity:** per-hop integrity tags prevent malleability and route tampering.

Threats (preview)

- End-to-end timing correlation.
- Malicious relays (guard/exit).
- Website fingerprinting.

[2] Indistinguishability under chosen-plaintext attack (e.g. AES-CBC with random IV, AES-CTR) ⇒ the relay cannot infer information about inner layers. In practice Tor uses AEAD (ChaCha20-Poly1305 or AES-GCM), which additionally provides per-hop integrity.

Onion Routing: Forward vs Backward Flow

Key reuse per hop

Each hop shares a single session key K_i with the client (derived during circuit build).

Forward path (client $\rightarrow$ server)

Client encrypts in order K_3, K_2, K_1 (innermost first); relays decrypt in order K_1, K_2, K_3.

Backward path (server $\rightarrow$ client)

Each relay *encrypts* with its K_i (reverse order overall). Client removes layers in order K_1, K_2, K_3.

Circuit State: How Relays Route Cells

- Relays do **not** inspect cell contents to make routing decisions.
- During circuit construction each relay stores a **circuit state entry**:

  $$\text{circuit ID} \;\mapsto\; (R_{i-1},\; R_{i+1},\; K_i)$$

- Every cell carries its circuit ID; the relay looks it up and forwards to R_{i+1} (forward path: decrypt with K_i), or to R_{i-1} (backward path: encrypt with K_i).

Onion Routing

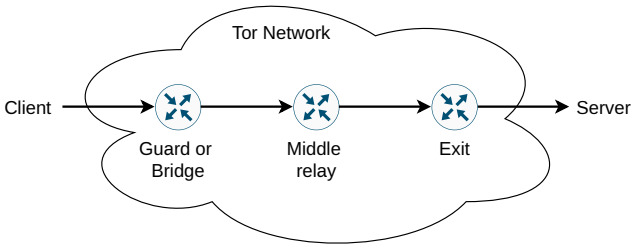
Tor
- Tor Relays and circuits
- Security properties and attacks
- Comments and future directions

Tor: Big Picture

- **Onion Proxy (Client)**: builds circuits; multiplexes application streams.
- **Relays**: *Guard* (entry), *Middle*, *Exit*.
- **Directory system**: small, hard-coded authorities + distributed consensus (relay keys, flags, bandwidth).
- **Circuits**: 3-hop paths (default); per-hop symmetric keys; layered encryption.
- **Cells**: fixed-size units on Tor links; circuits carry multiple TCP streams (e.g., several HTTPS tabs).

Tor: (The Onion Routing) Project

https://www.torproject.org/



Tor **circuit**:

- random path of at least 3 onion routers (one or more middle relays),
- using layered encryption.

Onion Routing

Tor
Tor Relays and circuits
Security properties and attacks
Comments and future directions

Relays: Guard, Middle, Exit

- **Guard**: first hop; sees client's IP; chosen from stable/high-quality relays to reduce exposure.
- **Middle**: transit only; no direct view of client or destination.
- **Exit**: last hop; connects to destination; subject to exit policies and abuse handling.

Design intuition

No single relay should learn both endpoints. Guard stability reduces the chance that the first hop is adversarial among many circuits.

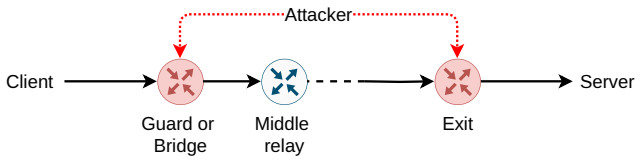
Guard node

Guard node: first node of a circuit.

- Required to have high uptime and minimum bandwidth.
- Each client chooses a Guard and uses it during a long time ($\sim$ 4 months).
- This improves resistance to some correlation attacks:
  - Predecessor attack: attacker controls first and last node of the circuit.
  - Denial of service as denial of anonymity: attacker forces the reset of circuits as middle relay to increase probability of predecessor attack.

Predecessor attacks in Tor

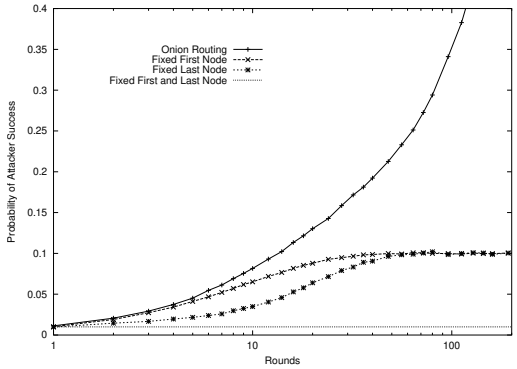
Attacker controls c nodes from total n nodes in the network. Best case for the attacker:



- Probability of attack success for one circuit (assuming uniform relay selection): $p \approx \left(\frac{c}{n}\right)^2$
- Probability of attack succes for k circuits: $\approx 1 - (1 - (\frac{c}{n})^2)^k \to 1$

Predecessor attack in Tor over time

- Without using a Guard, probability of predecessor attack increases over time.



Source: [4]

Directory and Consensus

- **Directory Authorities**: small, well-known set; relay descriptors feed a signed *consensus*.
- **Consensus**: network-wide snapshot (relay keys, addresses, flags, bandwidth weights).
- **Clients** fetch consensus periodically to select relays and build circuits.

Why consensus?

Ensures all honest clients have a consistent view of available relays, resisting partitioning and equivocation.

Tor directory authorities

Network information is handled by the **directory authorities**

- Maintain the list of Tor relays (including their public keys)
- Run by trusted individuals or organizations
- Authorities use a **consensus** protocol to share and maintain up to date information of the network.
- Goal:
  - Distribute critical information.
  - Avoid problems if an authority is compromised,
  - or avoid having a single point of attack/failure.

Onion Routing

Tor
Tor Relays and circuits
Security properties and attacks
Comments and future directions

Security Goals (Tor / Onion Routing)

- **Unlinkability**: adversary should not learn sender–receiver linkage.
- **Location privacy**: hide client IP from destination and exit; hide destination from guard.
- **Perfect Forward Secrecy**: fresh keys per hop/circuit (compromise now $\not\Rightarrow$ past traffic).
- **Integrity**: detect tampering at the link/circuit layer.

Non-goals / limitations (by design)

- Application-layer identity leakage (cookies, accounts, browser fingerprint) is out of scope.
- Global, perfectly synchronized passive adversary remains challenging.

End-to-End Correlation (Traffic Confirmation)

- Observe $\{x_t\}$: packet counts/timing at **entry (guard link)**.
- Observe $\{y_t\}$: packet counts/timing at **exit (dest link)**.
- Compute similarity (e.g., Pearson r, cross-correlation, DTW).
- High correlation $\Rightarrow$ candidate linkage.

Defenses

- *Circuit rotation* and *stream isolation*.
- *Padding* (link/circuit) and *traffic shaping*.
- *Path diversity* (guards, AS-aware selection).

Malicious Relays & Sybils

- **Guard compromise**: sees client IP; if also controls Exit ⇒ correlation risk.
- **Exit compromise**: can observe/modify cleartext to destinations lacking end-to-end TLS; can tag flows.
- **Sybil attacks**: attacker runs many relays to increase selection probability.

Mitigations

- Guard design: long-lived, vetted guard sets; slow rotation.
- Relay vetting, family/AS uniqueness, bandwidth weights.
- Exit policies, HTTPS-everywhere at application layer.

Network-Layer Attacks (BGP / Routing)

- **BGP hijacks/leaks**: steer traffic from/to guards/exits through adversarial ASes.
- **IX/AS correlation**: large IXPs or Tier-1 ASes observe both ends of many circuits.

Mitigations

- AS/IX-aware path selection; avoid shared AS on entry/exit.
- Monitoring for route anomalies.

Active Tagging & Watermarking

- Adversary perturbs timing/size patterns at entry (or exit) to detect them downstream.
- Works even with noise if watermark persists across the path.

Mitigations

- Per-hop integrity + rate limiting of unusual bursts.
- Padding and batching at relays reduce detectability.
- Circuit rebuild upon anomaly detection.

Client-Side Leaks & DNS Issues

- Browser / OS leaks: cookies, WebRTC/STUN, fonts, timezones, canvas, TLS session tickets.
- **DNS at Exit**: destination hostname leakage when apps resolve DNS locally (bypassing Tor's exit proxy) instead of delegating resolution to the exit relay; independent of HTTPS use.

Mitigations

- Tor Browser hardening; first-party isolation; new-circuit-for-site.
- Strict SOCKS use; disable proxy bypass; prefer HTTPS.
- Avoid logins with real identity when testing anonymity.

Defenses: Padding & Shaping (Trade-offs)

- Link vs circuit vs application-layer padding: different goals, overhead.
- Cover traffic improves anonymity but increases latency/bandwidth.
- Adaptive schemes aim to add noise where it helps correlation resistance most.

Design tension

More cover $\Rightarrow$ better correlation resistance;
too much cover $\Rightarrow$ unusable performance.

Summary of Attacks & Defenses

Attack	Signal	Assumption	Mitigation
End-to-end corr.	timing/volume	entry+exit vis.	padding, AS-aware, rotation
WF	burst shapes	stable site patterns	adaptive padding, shaping
Malicious guards	client IP	guard control	vetted guards, slow rotation
Malicious exits	dest traffic	exit control	HTTPS, exit policies
BGP/IX corr.	shared AS/IX	route control	AS-aware selection, RPKI
Watermarking	persistent pattern	active adversary	integrity, anomaly rebuild
HS targeting	intro/RP timing	service opsec	multi-IP, padding, rotation

Onion Routing

Tor
Tor Relays and circuits
Security properties and attacks
Comments and future directions

Performance Challenges

- **Low latency** is crucial for usability, but anonymity prefers delay and noise.
- Tor must balance:
  - latency vs. cover traffic,
  - throughput vs. congestion fairness,
  - performance vs. anonymity guarantees.
- Circuits share relay bandwidth; bottlenecks dominate end-to-end delay.

$\rightarrow$ Most performance issues stem from *network bottlenecks and congestion*, not cryptography.

Padding & Timing Defenses

- **Link padding:** maintain background traffic even when idle.
- **Circuit padding:** add dummy cells during activity bursts (e.g., WTF-PAD, Walkie-Talkie).
- **Adaptive schemes:** learn patterns online and add noise only when needed.
- Padding reduces correlation accuracy but adds measurable bandwidth cost.

Trade-off

Anonymity gain $\propto$ entropy increase, Cost $\propto$ extra bandwidth

Pluggable Transports (Censorship Resistance)

- Hide Tor traffic from network censors performing DPI or SNI filtering.
- **obfs4**: randomizes packet flow; resists active probing.
- **meek**: tunnels Tor over HTTPS through CDN domains.
- **snowflake**: WebRTC peer relays from volunteer browsers.
- **Bridges**: unlisted Tor relays not in the public consensus; pluggable transports typically run on top of bridges.

→ You can try it in our university (censored) network!!

Performance Measurements (Empirical)

- Median Tor RTT ≈ 200–400 ms; throughput ≈ 1–5 Mbit/s (2025 metrics).
- 90% page-load times ≈ 2× regular HTTPS.
- Bottlenecks: exit bandwidth, TLS handshake latency, TCP congestion.
- Active measurement platforms: `Tor Metrics`.

Summary & Takeaways

- Onion routing = layered encryption → unlinkability.
- Tor = practical deployment: 3-hop circuits, directory consensus, per-hop keys.
- Strong protection against local adversaries; weaker vs. global correlation.
- Enhancements focus on padding, congestion, censorship resistance.
- Future: mixnet hybrids (Loopix, Nym) and formal anonymity proofs.

Recall:
"If you have enough metadata, you don't really need content".
(Stewart Baker, former NSA General Counsel)
(privacy is not only about encryption, it's about hiding who talks to whom).

References i

[1] Marc Juárez Miró.
Protocols de comunicació que preserven la privadesa.
Fundació Universitat Oberta de Catalunya, 2020.

[2] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag.
Anonymous connections and onion routing.
Selected Areas in Communications, IEEE Journal on, 16(4):482–494, 1998.

[3] Vicenç Torra.
Guide to Data Privacy: Models, Technologies, Solutions.
Undergraduate Topics in Computer Science. Springer International Publishing, Cham, 2022.

[4] M. Wright, M. Adler, B.N. Levine, and C. Shields.
Defending anonymous communications against passive logging attacks.
In *2003 Symposium on Security and Privacy, 2003.*, pages 28–41, May 2003.

©G. Navarro-Arribas – (2026)