# NETHR: AN ANONYMOUS AND DECENTRALIZED NEXT-GENERATION P2P PROTOCOL

LAZAR ALEKSIĆ

Abstract. This paper presents the design, implementation, and empirical evaluation of Nethr, a next-generation decentralized P2P protocol engineered for resilient communication under active censorship and complete infrastructure collapse. The research addresses the fundamental weaknesses of existing privacy-preserving systems: their systemic reliance on central infrastructure and their susceptibility to protocol fingerprinting. We introduce three novel mechanisms that, in combination, provide a level of unobservability not achieved by existing protocols: symmetric trial decryption, identifierless deduplication, and native wire-format indistinguishability. Our methodology utilizes a custom C++ simulation engine to evaluate Bounded Epidemic Routing across hybrid scale-free network topologies, alongside rigorous statistical analysis of live-captured Nethr traffic. Results demonstrate that Nethr sustains a message delivery rate above 90% at fanout k = 3 even under 40% network failure, rising to above 99% at k = 5 under 40% failure. Crucially, we observe that Nethr's delivery remains robust under sophisticated targeted attacks on high-degree network hubs. A result that counter-intuitively suggests that hubs act as routing bottlenecks rather than amplifiers in a bounded epidemic model. Statistical analysis via Chi-square uniformity, Kolmogorov-Smirnov, Shannon entropy, and Pearson autocorrelation tests proves that Nethr traffic is mathematically indistinguishable from uniform random noise across the entire packet, including the header. By utilizing a hybrid post-quantum key exchange (Noise_XX with sntrup761/X25519) and authenticated symmetric encryption (ChaCha20-Poly1305), Nethr delivers quantum-resistant security and low convergence latency on commodity hardware without any reliance on central authorities.

Keywords: anonymous communication, decentralized protocols, post-quantum cryptography, deep packet inspection, epidemic routing, censorship resistance, unobservability, traffic analysis resistance.

## 1. Introduction

The internet is increasingly neither universal nor free. In 2025 alone, the Access Now KeepItOn coalition documented 313 internet shutdowns across 52 countries. State-level administrative disruptions of connectivity utilized as mechanisms for infrastructure-wide information control [1]. Freedom House's Freedom on the Net report recorded the fifteenth consecutive year of declining global internet freedom, with deterioration outpacing improvement by a ratio of nearly two to one [2]. These trends are not incidental. Network authorities have progressively augmented passive surveillance with active interdiction: deep packet inspection (DPI) systems that classify and block traffic based on protocol fingerprinting, automated filtering at autonomous system boundaries, and infrastructure-level blocking of known anonymity tools [3]. The result is a systematic erosion of the conditions under which private communication is possible.

*Date*: June 14, 2026.

Existing systems attack this problem from different angles, and each reaches a different wall. Tor and I2P provide meaningful anonymity against passive observers, but both depend on discoverable infrastructure-directory authorities and reseed servers-that an adversary can block without touching any cryptographic primitive. Bitmessage sidesteps infrastructure entirely by flooding every message to every participant, but transmits deduplication identifiers in plaintext and its bandwidth requirements make it impractical at scale. Mix network systems like Vuvuzela and Loopix offer the strongest formal anonymity guarantees in the literature, but through central servers that must be trusted and can be blocked, at latency costs of 37 seconds in Vuvuzela's case, that rule out real-time use. EFPIX [10] improves on the Bitmessage model but remains theoretical. None of these systems solve a more fundamental problem: their traffic is identifiable as anonymity-system traffic by a DPI classifier. An infrastructure adversary does not need to break their cryptography; it only needs to recognize their packets and drop them. This paper presents Nethr, a next-generation decentralized peer-to-peer communication protocol designed to operate under active infrastructure-level censorship and the complete absence of trusted third parties. Nethr's design is organized around a single core insight, that metadata protection and protocol unobservability must be achieved at the packet level, not merely at the routing level, because an infrastructure adversary who can identify a protocol's traffic can block it regardless of the strength of its cryptographic guarantees. From this principle follow three novel mechanisms, symmetric trial decryption, identifierless deduplication, and BLAKE3-based header masking, that collectively eliminate all statically observable metadata from Nethr traffic. The resulting wire format is computationally indistinguishable from uniform random noise across all 1472 bytes of every packet, verified empirically through four independent statistical tests on live-captured traffic. Nethr routes messages using a bounded epidemic algorithm over a hybrid Barabási-Albert scale-free topology, providing delivery robustness without routing tables, directory services, or any persistent state beyond a local peer cache. Session keys are established through a Noise_XX handshake with a hybrid post-quantum construction combining X25519 and sntrup761, delivering both classical and post-quantum forward secrecy. The entire protocol operates without any central infrastructure: no directory servers, no mix nodes, no service providers, and no bootstrap authorities beyond peer introduction.
This paper makes the following contributions:

(1) **Protocol design:** We present Nethr, a decentralized P2P protocol that simultaneously achieves serverless operation, DPI-resistant wire format, post-quantum forward secrecy, and metadata protection without the latency penalties of traditional mix networks.
(2) **Three novel mechanisms:** We describe *symmetric trial decryption*, which achieves recipient anonymity without explicit identifiers; *identifierless deduplication*, which eliminates message tracking across relays; and *BLAKE3-masked headers*, which produce a wire format statistically indistinguishable from random noise.
(3) **High-Assurance Key Exchange Integration:** We implement a hybrid X25519 + sntrup761 key exchange synthesized within the Noise_XX handshake framework. While this cryptographic combination is utilized in client-server environments like OpenSSH, we demonstrate its application in a fully decentralized, epidemic-routing context to provide classical and post-quantum forward secrecy without relying on central bootstrap authorities.
(4) **Empirical evaluation:** We evaluate Nethr via a production-grade C++ daemon and simulation engine. Results show message delivery above 90% under 40% node failure at fanout $k = 3$, while statistical tests confirm wire-format indistinguishability from uniform noise.

(5) **The Hub-Bottleneck Effect:** We formally explain a counter-intuitive topology result where targeted removal of high-degree hubs degrades delivery less than random node failure, challenging classical assumptions about scale-free network vulnerability.

Section 2 surveys related work and identifies the specific limitations motivating Nethr's design. Section 3 defines the threat model and states security goals. Section 4 describes the protocol design in detail. Section 5 analyzes Nethr's security properties. Section 6 describes the implementation. Section 7 presents the experimental evaluation. Section 8 discusses design tradeoffs and limitations. Section 9 concludes.

## 2. Related Work

The privacy and censorship-resistance literature has produced four distinct architectural approaches to anonymous communication. Each solves part of the problem Nethr addresses; none solves all of it. We survey the most relevant prior work and identify precisely where each approach falls short against an infrastructure-level adversary.

### 2.1. **Onion Routing and Circuit-Based Systems.**

Onion routing, introduced by Goldschlag et al. [4] and refined into the Tor network by Dingledine et al. [5], remains the most widely deployed anonymous communication system. Tor establishes layered encrypted circuits through a sequence of three or more volunteer-operated relay nodes. Each relay decrypts one layer of encryption, learning only its predecessor and successor in the circuit, thereby providing sender anonymity through indirection. Hidden services extend this model to protect server identity through rendezvous-point circuits.

Tor's primary vulnerability in our threat model is its dependence on a set of nine hardcoded directory authority servers that maintain consensus over the list of active relays. An infrastructure adversary controlling the autonomous systems hosting these IP addresses can prevent new clients from bootstrapping into the network entirely. This is a practical concern; documented censorship events in several regions have exploited this infrastructure dependency [3]. While Tor's pluggable transport system (e.g., obfs4, Snowflake) addresses DPI fingerprinting, these transports are not enabled by default, require explicit configuration, are not part of the protocol design, and remain susceptible to identification via flow-feature analysis. Researchers have demonstrated that even obfuscated transports like obfs4 exhibit unique statistical signatures in their packet length distributions and timing patterns, allowing machine learning classifiers to distinguish them from "unstructured" background traffic with high confidence [6].

The fundamental architecture of Tor: centralized relay directories, fixed circuit topologies, and distinct handshake patterns, create multiple surfaces for infrastructure-level identification that Nethr's design seeks to eliminate.

The Invisible Internet Project (I2P) [7] addresses directory centralization through a distributed network database (NetDB) maintained across volunteer floodfill routers using a Kademlia-based DHT. While architecturally more resistant to blocking than Tor, I2P still requires reseed servers for initial bootstrap. A censor blocking these server addresses prevents new I2P nodes from joining the network. Furthermore, while I2P deployed hybrid ML-KEM + X25519 encryption in 2026 [8], its wire format remains distinguishable from random noise at the packet level. Without header obfuscation equivalent to Nethr's BLAKE3-based masking, I2P traffic remains susceptible to DPI-based protocol fingerprinting.

### 2.2. **Broadcast and Flood-Based Systems.**

An alternative approach to recipient anonymity eliminates routing entirely: every message is distributed to every network participant, and each participant attempts decryption. Bitmessage [9] implements this model using

asymmetric ECIES encryption. While this achieves genuine recipient anonymity, Bitmessage transmits a SHA-512 hash of the message payload as an unencrypted field for deduplication. This constitutes a metadata leak, allowing an observer to track message propagation across the network. Furthermore, Bitmessage's full-flood model scales poorly; bandwidth consumption grows as O(N×M) for N nodes and M messages, rendering the network impractical at scale. Nethr utilizes bounded epidemic routing to provide tunable delivery probability without the overhead of full-network flooding.

2.3. **Mix Networks and Differential Privacy Systems.** Mix networks [11] achieve anonymity through batching and reordering. Vuvuzela [12] represents the state of the art in formally proven anonymous messaging, providing differential privacy guarantees ($\epsilon$, $\delta$). However, this comes at a substantial latency cost, approximately 37 seconds at scale, and relies on central servers that must be trusted and are easily blocked. Loopix [13] addresses this latency problem using Poisson-distributed message delays, achieving latency on the order of seconds. While Loopix is the closest prior work to Nethr in its cover traffic approach, it operates through a stratified topology of mix nodes and service providers. A service provider that is blocked or compromised degrades anonymity for all users who route through it. Nethr's cover traffic operates at the individual node level, eliminating any infrastructure dependency.

2.4. **Post-Quantum Cryptography in Network Protocols.** The threat of cryptographically-relevant quantum computers to public-key infrastructure has motivated the standardization of lattice-based algorithms, most notably ML-KEM (FIPS 203) [14]. However, Nethr utilizes *sntrup761* (Streamlined NTRU Prime) [15] as its post-quantum KEM. This choice is motivated by two factors. First, the NTRU lattice problem has resisted cryptanalysis for nearly three decades (since 1996), providing a more conservative and "battle-tested" security foundation than newer module-lattice constructions. Second, *sntrup761* follows a "Nothing Up My Sleeves" design philosophy, utilizing a structure that minimizes the use of suspicious constants or algebraic shortcuts. This addresses growing community skepticism regarding potential structural vulnerabilities or "backdoor" potential in NIST-standardized algorithms. The deployment of *sntrup761* in OpenSSH since 2021 [16] provides additional empirical evidence of its implementation quality and performance on commodity hardware. By employing a hybrid X25519 + *sntrup761* construction, Nethr ensures that session traffic remains secure as long as at least one of the two primitives is not compromised. This provides a robust defense against "harvest now, decrypt later" attacks without relying on a single, recently-standardized cryptographic assumption.

2.5. **The Gap.** No surveyed system simultaneously achieves serverless operation, DPI-resistant wire format, post-quantum security, low latency, and identifierless deduplication. Table 1 makes this comparison explicit. The design space defined below establishes the constraints Nethr is built to satisfy.

## 3. Threat Model

We define the adversarial model against which Nethr is designed and evaluated. This model considers the capabilities of various entities to observe, modify, or disrupt network traffic, and establishes the boundaries of the protocol's guarantees.

TABLE 1. Comparison with related anonymous communication protocols

| Feature | Tor | I2P | Bitmessage | Vuvuzela | Nethr |
|---|---|---|---|---|---|
| Serverless Operation | × | ∼ | ✓ | × | ✓ |
| DPI Resistance (Native) | × | × | × | × | ✓ |
| Post-Quantum Security | ∼ | ✓ | × | × | ✓ |
| Low Latency (<1s) | ✓ | ✓ | × | × | ✓ |
| Metadata Protection | ✓ | ✓ | ∼ | ✓ | ✓ |
| Identifierless Deduplication | N/A | N/A | × | N/A | ✓ |

3.1. **Network Model.** Nethr operates over a permissionless, fully decentralized P2P network. Nodes communicate exclusively over UDP/IP. No trusted third parties or directory servers exist. The network graph is dynamic and evolves according to the Barabási-Albert preferential attachment mechanism, producing a scale-free topology with small-world properties. All nodes are assumed to be reachable via standard IP routing, though the protocol is designed to remain operational under partial connectivity. We acknowledge that like all P2P protocols, the participant set is not entirely unobservable; an adversary with sufficient resources can potentially graph the network by observing connection handshakes.

3.2. **Adversary Classes. The Infrastructure Adversary (IA)** is Nethr's primary design target. The IA controls network infrastructure: routers, autonomous systems, internet exchange points, and ISP-level equipment. IA can observe, filter, throttle, block, or redirect traffic based on any observable packet characteristic. This adversary can perform deep packet inspection (DPI) on all traffic transiting its infrastructure and is assumed to have the power to compromise or block centralized entities, such as directory authorities or reseed servers. While the IA may attempt and succeed in targeted endpoint captures or the exploitation of vulnerabilities in classical cryptographic primitives, such attacks are ineffective for large-scale deanonymization due to Nethr's decentralized architecture and the integration of post-quantum cryptographic primitives. This adversary class models nation-state censors and large-scale network surveillance operators whose distinguishing capability is the power to block traffic based on protocol fingerprinting, even in cases where content decryption remains infeasible.

**The Global Passive Adversary (GPA)** can observe all traffic on all network links simultaneously. The GPA records every packet, its source, destination, timing, and size across the entire network, but does not modify traffic or inject packets. The GPA can perform statistical traffic analysis and timing correlation across all observed flows. This represents the theoretical upper bound on passive surveillance capability.

**The Local Active Adversary (LAA)** controls a fraction f of network nodes. These nodes may be Sybil identities or a coalition of colluding malicious participants. LAA-controlled nodes may refuse to forward packets, selectively drop traffic, send false peer exchange information, or attempt to correlate traffic by controlling routing paths. The LAA cannot break cryptographic primitives and cannot observe traffic on links between honest nodes. Our experiments evaluate LAA resistance for $f \in \{0, 0.1, 0.2, 0.3, 0.4\}$.

**The Global Active Adversary (GAA)** possesses the capabilities of the GPA but can also inject, modify, reroute, or suppress traffic globally. Unlike circuit-based systems, Nethr is inherently resilient to many classical GAA attacks. Cryptographic modification or injection is prevented by the ChaCha20-Poly1305 AEAD and BLAKE3 MAC, which ensure that altered or unauthorized packets are dropped during the unmasking and verification phase. Furthermore,

since Nethr utilizes stateless epidemic routing rather than fixed circuits, active rerouting attacks are ineffective; the protocol's redundancy ensures that the message continues to propagate even if specific flows are diverted. The primary threat posed by a GAA remains massive-scale packet suppression or long-term intersection analysis across the entire internet backbone.

3.3. **Security Goals.** Against the IA, Nethr delivers the following guarantees:

- **Protocol unobservability:** No Nethr packet contains any static byte pattern, magic number, or detectable structure. Every packet is computationally indistinguishable from uniform random noise across all 1472 bytes, including the header. This is achieved through BLAKE3-based header masking and ChaCha20-Poly1305 encryption of the body.
- **Content confidentiality:** Message content is protected by ChaCha20-Poly1305 AEAD encryption under a session key established through Noise_XX key exchange. An IA that records all traffic cannot decrypt message content without the session key.
- **Length anonymity:** All packets are currently padded to exactly 1472 bytes regardless of message length, eliminating packet-size side channels.

Against the GPA and LAA, Nethr establishes the following:

- **Probabilistic Sender Anonymity:** While an originating node is observable to its k immediate neighbors, Nethr provides a layer of probabilistic anonymity. Because every node performs identical trial-decryption and forwarding logic, a neighbor cannot distinguish with certainty whether a node is the message origin or merely a relay. Furthermore, the continuous generation of cover traffic ensures that a neighbor cannot be certain if a specific packet represents a real message exchange or background network maintenance.
- **Recipient anonymity:** No identifier, address, or token identifying the intended recipient is transmitted. A GPA observing packets in transit cannot determine the intended recipient; every node that receives and forwards a packet is observationally indistinguishable from the actual recipient.
- **Forward secrecy:** Session keys are derived from ephemeral Diffie-Hellman values generated fresh for each Noise_XX handshake. Long-term static key compromise does not expose past session traffic.
- **Post-quantum forward secrecy:** The hybrid X25519 + sntrup761 KEM construction ensures that an adversary recording traffic today cannot retroactively decrypt it after acquiring quantum computing capability, provided at least one of the two KEM components remains secure.
- **Timing metadata resistance:** The dynamic cover traffic mechanism generates cryptographically indistinguishable cover packets at continuously varying random intervals, preventing a GPA from correlating message transmission and reception times through simple timing analysis.

Against the LAA specifically:

- **Delivery robustness under node failure:** The bounded epidemic routing algorithm with fanout k provides redundant delivery paths. At $k = 3$, the protocol maintains $\geq 90\%$ message delivery under f=0.4 random node failure.
- **Sybil infiltration resistance:** The Barab'asi-Albert preferential attachment topology structurally marginalizes newly introduced Sybil nodes to the network periphery, limiting their ability to occupy high-centrality routing positions.

Against the GAA, Nethr provides non-trivial resilience against active network-level manipulation:

- **Integrity and Authenticity:** The combination of ChaCha20-Poly1305 AEAD and the BLAKE3-masked MAC ensures that any attempt by a GAA to modify packets in transit or inject unauthorized traffic results in immediate validation failure. Because the unmasking process is cryptographically bound to the body, modified packets cannot be successfully processed by relays or the intended recipient.
- **Resilience to Rerouting and Diversion:** Unlike circuit-based protocols (e.g., Tor or I2P) where a GAA can attempt to divert a specific circuit through adversary-controlled nodes, Nethr's stateless epidemic routing makes rerouting attacks ineffective. Since there is no fixed path to "capture," and messages propagate via k redundant branches simultaneously, the protocol naturally routes around active interdiction and diverted flows.
- **Active Replay Protection:** The identifierless deduplication mechanism ensures that even if a GAA captures and re-injects packets into the network to perform timing analysis or resource exhaustion, these packets are recognized as duplicates and dropped at the first honest hop, preventing the adversary from triggering repeated network-wide propagation.

3.4. **Explicit Non-Goals and Future Work.** Nethr does not currently provide the following properties:

- **Participant set unobservability:** An IA controlling network infrastructure can potentially detect the existence of nodes running Nethr. It is important to note that this is not a failure of the packet format, which remains statistically indistinguishable from noise, but rather an inherent challenge of decentralized peer-to-peer systems. An adversary can attempt to graph the network by correlating UDP flow timing and peer-to-peer handshakes between IP addresses. This "graphing problem" is common to all P2P protocols that actively exchange packets and keys.
- **Variable-Size Obfuscation:** While Nethr currently utilizes a fixed 1472-byte datagram to eliminate length-based side channels, future work will explore *randomizable* packet sizes. This will involve the use of dynamic padding to frustrate sophisticated statistical classification attempts that might target the consistency of the 1472-byte wire format.
- **Formal differential privacy:** Unlike Vuvuzela [12], Nethr does not currently provide mathematically proven differential privacy bounds. Establishing formal differential privacy guarantees for bounded epidemic routing remains a primary area for future research.
- **Resistance to endpoint compromise:** Nethr assumes a secure execution environment. If an adversary successfully compromises a node's device, all session keys and plaintext are exposed. Future work will investigate the integration of local encryption and destruction, hardware security modules (HSM) and TEE-based isolation to mitigate this risk.

## 4. Protocol Design

Nethr's design is organized around a single architectural invariant: every byte visible to a network observer must be computationally indistinguishable from uniform random noise. All

protocol mechanisms such as session establishment, message construction, forwarding, deduplication, and cover traffic, are subordinate to this requirement.

4.1. **System Architecture.** The Nethr protocol is implemented as a standalone C++20 daemon process. The daemon owns all network I/O, cryptographic state, and routing logic. All external interaction comes from the Flutter graphical interface and from Python experiment automation scripts. This occurs exclusively through a TCP inter-process communication (IPC) server bound to `127.0.0.1:9090`. The IPC server accepts a persistent connection and exchanges JSON-lines messages: the client sends commands, the daemon responds immediately, and the daemon may push unsolicited events at any time without polling.

This strict separation ensures that the protocol's network-facing behavior is fully determined by the daemon and cannot be inadvertently affected by GUI state or test script logic. The IPC channel is loopback-only and carries no cryptographic material; all session keys and peer identities are managed exclusively within the daemon process.

The daemon operates four concurrent logical threads. The main thread manages daemon lifecycle, initiates outbound handshakes, and dispatches IPC commands. A UDP receive thread runs a tight `recvfrom()` loop, parses and authenticates incoming packets, and dispatches them to the router. The IPC server thread accepts client connections and spawns a reader/writer thread pair per client, supporting server-initiated push events. A timer thread periodically triggers cover traffic generation, keepalive transmission, session expiry, and peer exchange. Handshakes are handled by short-lived threads spawned on demand, decoupled from the main receive loop through a per-peer packet queue inside the router.

Cryptographic operations are isolated in a dedicated `crypto::` namespace that provides typed wrappers around libsodium 1.0.21 [19] and liboqs 0.15.0 [20]. No other module calls libsodium or liboqs symbols directly, enforcing a single auditable cryptographic boundary.

4.2. **Wire Format.** Every Nethr packet transmitted on the network is exactly **1472 bytes**. This is the maximum UDP payload that traverses standard Ethernet without IP fragmentation: $1500 - 20$ (IPv4) $- 8$ (UDP) $= 1472$ bytes. Fixed packet size eliminates length-based side channels entirely.

The wire format consists of a 48-byte header followed by a 1424-byte body. Table 2 describes the header field layout.

**Header masking.** Before transmission, the header is XOR-masked using the first 48 bytes of:

$$\text{mask} = \text{BLAKE3}(\texttt{"header-mask-v2"} \parallel \text{body})[0..47]$$

Since the body is always a ChaCha20-Poly1305 ciphertext or cryptographic random padding, both computationally indistinguishable from uniform random, the derived mask is a fresh, independent 48-byte value for every packet.

XOR with a uniform random mask produces a uniform random output regardless of the input structure, eliminating all static byte patterns including magic bytes, version, and type field. XOR is its own inverse, the same operation unmasks on receipt. No static byte pattern survives masking across any pair of packets.

**MAC computation.** The 16-byte BLAKE3 MAC covers the unmasked header fields at bytes 0–7 (comprising the `magic` identifier, `version`, `type`, `payload_len`, `ttl`, and `flags`), the 12-byte nonce at bytes 28–39, and the full 1424-byte body:

$$\text{MAC} = \text{BLAKE3}(\,\text{hdr}[0..7] \parallel \text{hdr}[28..39] \parallel \text{body}\,)\,[0..15]$$

The MAC is computed over the raw unmasked header before transmission and verified immediately after unmasking on receipt. Comparison uses `sodium_memcmp()` to prevent timing side

TABLE 2. Nethr packet header layout (48 bytes, transmitted masked)

| Offset | Bytes | Field | Value / Constraint |
|---|---|---|---|
| 0 | 1 | `magic[0]` | `0x4F` - fixed |
| 1 | 1 | `magic[1]` | `0x50` - fixed |
| 2 | 1 | `version` | `0x01` - fixed |
| 3 | 1 | `type` | 0=Handshake, 1=Message, 2=Keepalive, 3=Bootstrap, 4=PEX |
| 4 | 2 | `payload_len` | LE uint16; ciphertext byte count; max 1424 |
| 6 | 1 | `ttl` | Initial value 8; decremented before forwarding; drop at 0 |
| 7 | 1 | `flags` | Always `0x00` |
| 8 | 4 | `reserved1` | Always `0x00000000` |
| 12 | 16 | `mac` | BLAKE3 authentication tag (see below) |
| 28 | 12 | `nonce` | Random per Message/PEX; zero for all other types |
| 40 | 8 | `reserved2` | Always zero |
| 48 | 1424 | `body` | Ciphertext (`payload_len` bytes) + random padding |

channels. Any single-bit modification to any field in the 8-byte header prefix, the nonce, or any byte of the body causes the MAC to fail, and the packet is silently dropped. The MAC is computed over the raw unmasked header before transmission and verified immediately after unmasking on receipt. Comparison uses `sodium_memcmp()` to prevent timing side channels. Any single-bit modification to the TTL, type, payload length, nonce, or any byte of the body causes the MAC to fail, and the packet is silently dropped.

**Deduplication key.** The deduplication key is BLAKE3(body)[0..15], computed locally from the encrypted body and never transmitted. Because the body bytes are not modified as a packet propagates, only the TTL in the header changes, the key is stable across all hops, allowing any node to recognize a previously seen packet without any transmitted identifier.

4.3. **Session Establishment.** Session establishment uses the Noise_XX handshake pattern [18], instantiated as `Noise_XX_25519_ChaChaPoly_BLAKE2s`. The XX pattern is chosen because it requires no prior knowledge of the peer's static key, enabling session establishment between any two previously unknown nodes.

The handshake proceeds in three acts, each fitting within a single 1472-byte Nethr Handshake packet. The sntrup761 KEM [15] runs in parallel with the Noise framework's X25519 DH exchange.

- **Act 1** (≈1190 bytes): The initiator generates a fresh ephemeral sntrup761 keypair for this handshake and transmits the noise-c ephemeral X25519 message (32 bytes) concatenated with the sntrup761 ephemeral public key (1158 bytes).

- **Act 2** (≈1135 bytes): The responder runs `OQS_KEM_encaps` using the initiator's sntrup761 public key, obtaining a 1039-byte ciphertext and shared secret $ss_{\text{pqkem}}$. The responder transmits the noise-c Act 2 message (≈96 bytes: ephemeral key, DH(ee), encrypted static key, DH(es)) concatenated with the sntrup761 ciphertext.
- **Act 3** (≈64 bytes): The initiator runs `OQS_KEM_decaps` using its ephemeral sntrup761 private key to recover $ss_{\text{pqkem}}$, then transmits the noise-c Act 3 message (encrypted static key, DH(se)).

All three acts fit within single 1472-byte packets; no fragmentation occurs. After completion, both parties derive session keys through two-step HKDF [17]:

$$\text{ikm} = H_{\text{noise}} \,\|\, ss_{\text{pqkem}} \qquad \text{prk} = \text{HKDF-Extract}(\text{salt} = 0^{32},\ \text{ikm})$$

where $H_{\text{noise}}$ is the 32-byte handshake transcript hash produced by `noise_handshakestate_get_handshake_hash()`. This hash cryptographically binds the resulting session keys to the entire handshake transcript, including all exchange messages and the three classical DH operations (*ee*, *es*, *se*). The *ikm* (Input Keying Material) thus incorporates both the authenticated classical transcript and the post-quantum shared secret.

Directional session keys are then derived using domain-separated labels:

$$tx_key_{\text{init}} = rx_key_{\text{resp}} = \text{HKDF-Expand}(\text{prk},\ \text{"tx-v1"},\ 32)$$

$$rx_key_{\text{init}} = tx_key_{\text{resp}} = \text{HKDF-Expand}(\text{prk},\ \text{"rx-v1"},\ 32)$$

where $L_1$ and $L_2$ are distinct domain separation labels for the two directions. The final session key is bound jointly to the classical X25519 transcript and the sntrup761 shared secret. Compromise of either component alone is insufficient to recover the session key, satisfying the standard hybrid KEM security argument.

Successfully established sessions are stored with their directional keys, peer identifier, and last-active timestamp. Sessions expire after 10 minutes of inactivity. Up to 64 peers are tracked in a fixed-size cache with LRU eviction.

4.4. **Message Lifecycle. Message construction.** When a user sends a message to a peer with an established session, the following pipeline executes:

(1) A fresh 12-byte nonce is generated via `randombytes_buf()`.
(2) An inner plaintext is formed by prepending a single `InnerType` byte (`0x01` for application messages) to the user's plaintext. This byte distinguishes message payloads from PEX payloads inside the ciphertext, invisible to any observer without the session key.
(3) The inner plaintext is encrypted: ct = ChaCha20-Poly1305-Encrypt($tx_key$, nonce, inner), producing a ciphertext of length |plaintext| + 1 + 16 bytes (the 16-byte Poly1305 tag is appended).
(4) The body is formed by placing the ciphertext at offset 0 and filling the remaining 1424 − |ct| bytes with `randombytes_buf()` output.
(5) The MAC is computed over the raw header fields and body.
(6) The header is XOR-masked with BLAKE3(`"header-mask-v2"` ‖ body)[0..47].
(7) The masked header and body are transmitted as a single 1472-byte datagram to $k$ randomly selected peers.

**Reception and trial decryption.** On receiving a 1472-byte datagram, a node executes the following procedure:

(1) Unmask the header: hdr = wire[0..47] ⊕ BLAKE3(`"header-mask-v2"` ‖ body)[0..47].

(2) Parse and validate structural constraints (magic bytes, version, reserved fields, type range).
(3) Verify the BLAKE3 MAC using `sodium_memcmp()`. Drop on failure.
(4) Compute dk = BLAKE3(body)[0..15]. If dk is present in the deduplication cache, drop silently.
(5) Insert dk into the deduplication cache. Drop if TTL = 0.
(6) For each session $s$ in the session table: attempt pt = ChaCha20-Poly1305-Decrypt($s.rx_key$, hdr.nonce, body[0..payload_len]). If the Poly1305 tag validates, deliver pt[1 :] to the application and break.
(7) If `hdr.type = Message` and `hdr.ttl > 1`: decrement TTL, recompute the MAC over the new header, remask the header, and transmit the packet to $k$ randomly selected peers (excluding the source).

The critical anonymity property is explicit in step 7: forwarding occurs unconditionally for all Message packets with TTL $> 1$, regardless of whether trial decryption succeeded. The recipient node's outgoing behavior is observationally identical to any relay node. An observer cannot distinguish a node that decrypted the message from one that simply forwarded it. The Poly1305 authentication tag serves as an implicit recipient identifier: it validates only for the node holding the correct session key, and its success or failure produces no externally observable signal.

The nonce is preserved unchanged during forwarding. Only the TTL is decremented and the MAC recomputed to reflect the new header state.

**Deduplication cache.** The cache holds up to 4096 entries in a FIFO queue with a parallel hash set for O(1) average-case lookup. When the cache is full, the oldest entry is evicted. The 16-byte deduplication key never leaves the device.

4.5. **Cover Traffic and Timing Protection.** When a node is communicating, its metadata is protected through a dynamic cover traffic mechanism. Traditional constant-rate dummy traffic imposes fixed bandwidth overhead; traditional low-rate dummy traffic produces high latency. Nethr's approach uses continuously varying random time windows to eliminate both problems.

A background thread maintains a dynamic time window of random duration that shifts continuously. At the expiration of each random delay sampled from within the current window, if no real message has been sent, a Keepalive packet (type `0x02`) is transmitted: a 1472-byte datagram with a zero nonce and a body filled entirely by `randombytes_buf()`. Because both Message and Keepalive bodies are computationally indistinguishable from uniform random noise, exactly as ChaCha20-Poly1305 ciphertext, a passive network observer operating on raw wire bytes cannot distinguish active communication from cover traffic through body analysis alone.

It should be noted, however, that because the current header mask is derived solely from the public body bytes,

$$\text{mask} = \text{BLAKE3(body)}[0..47],$$

an adversary who actively computes this value per received packet can invert the mask and recover the plaintext header, including the type field that distinguishes Message from Keepalive packets. This limits the current cover traffic guarantee to passive classifiers rather than computationally active adversaries. A potential mitigation involves generating fully authentic, session-bound headers for all cover packets to ensure bit-level identity with legitimate messages. However, this approach would impose high computational overhead on network participants, as

nodes would be required to perform continuous high-frequency cryptographic operations even during periods of network inactivity.

The protocol design assumes the implementation of a *keyed* BLAKE3 construction for header masking, where the masking key is derived directly from the Noise session secret:

$$\text{mask} = \text{BLAKE3}_{\text{key}}(\text{body})[0..47], \qquad \text{key} = \text{HKDF-Expand}(\text{prk}, \text{"nethr-mask-v1"}, 32)$$

By utilizing a key unknown to the network observer, the unobservability guarantee is extended from passive classifiers to active adversaries. An observer without the session key cannot compute the mask and therefore cannot invert the XOR operation to recover the plaintext header. This ensures that the unobservability property holds even against DPI systems capable of performing per-packet hash computations. For the purposes of the formal analysis in this paper, this keyed construction is considered the standard implementation.

This approach trades the strict Poisson-distribution model of systems like Loopix [13], which yields formal unlinkability bounds, for tighter latency control. Real messages replace the next scheduled cover packet rather than waiting for a fixed batching window, achieving minimal additional latency for legitimate communication. As stated in Section 3.4, the mechanism protects against local timing adversaries but does not prevent an infrastructure adversary from identifying the participant set by observing continuous fixed-size UDP traffic at irregular but consistent intervals.

4.6. **Network Topology and Peer Discovery.** The network topology is generated by combining a ring structure with the Barabási-Albert (BA) preferential attachment model [21]. All $N$ nodes are first connected in a ring, providing uniform baseline connectivity. For each node, preferential attachment then adds up to $k$ additional edges, selecting targets with probability proportional to their current degree. This produces a scale-free network whose degree distribution follows a power law $P(d) \sim d^{-\gamma}$ with $\gamma \approx 3$, with hub nodes that have disproportionately high connectivity alongside many low-degree peripheral nodes. The ring component introduces local clustering; the BA component reduces average path length, together yielding small-world properties [22].

Peer discovery in production operation uses two mechanisms. UDP broadcast on `INADDR_BROADCAST` allows a new node to announce its presence to all hosts on the same local subnet via Bootstrap packets (type `0x03`). For wide-area connectivity, peer addresses are loaded from a plaintext file or introduced manually through the IPC interface. Peer entries are stored in a fixed-size cache of 64 entries with LRU eviction. The Peer Exchange (PEX) mechanism periodically sends an encrypted PEX packet (type `0x04`) containing up to 10 known peer addresses to a randomly selected session peer, enabling organic network growth without a central directory. PEX payloads use the same AEAD encryption and header masking as Message packets and are therefore indistinguishable from them on the wire.

In the simulation environment, all sessions are pre-established using randomly generated 32-byte symmetric keys via `randombytes_buf()`, isolating the evaluation of epidemic routing and trial decryption performance from handshake overhead. This is the correct experimental design for evaluating the routing algorithm's properties independently. All $N$ nodes are wired through the BA topology construction, and $N(N-1)/2$ mock session pairs are established before any message injection begins.

## 5. Security Analysis

The three adversary classes defined in Section 3 require fundamentally different arguments. Against the infrastructure adversary, the central question is computational: does the protocol produce distinguishable traffic? Against the global passive adversary, the question is information-theoretic: what does observing all packets reveal about communication relationships? Against the local active adversary, the question is probabilistic: how much of the routing graph must an adversary control before delivery meaningfully degrades? We address each in turn, stating explicit claims, providing security arguments grounded in standard cryptographic assumptions, and identifying where each guarantee ends.

5.1. **Protocol Unobservability Against the Infrastructure Adversary. Claim:** No Nethr packet contains any statistically detectable structure that an IA can use to distinguish it from uniform random noise.

**Argument:** Consider a Nethr packet $P = H \parallel B$, where $H$ is the 48-byte header and $B$ is the 1424-byte body. The body $B$ is the output of ChaCha20-Poly1305 encryption of the inner message. Under the standard security definition of AEAD encryption, $B$ is computationally indistinguishable from uniform random under a chosen-plaintext attack (IND-CPA). The transmitted header $H$ is computed as $H = H_{\text{raw}} \oplus \text{BLAKE3}_k(B)[0..47]$, where $k$ is the session-specific masking key. Because $k$ is derived from the Noise handshake and remains unknown to the observer, the output of the keyed BLAKE3 function is computationally indistinguishable from a random string (treating the keyed hash as a Pseudorandom Function, or PRF). Since XORing a structured input ($H_{\text{raw}}$) with a PRF output produces a result that is computationally indistinguishable from uniform random, $H$ is also indistinguishable from noise. Therefore, an efficient adversary holding $P$ and $B$ cannot distinguish the resulting datagram from random noise without knowledge of $k$.

**Empirical confirmation:** The theoretical argument is confirmed empirically in Section 7. Chi-square uniformity tests yield p-values approximating 0.5 for the Nethr header. Shannon entropy is ∼8 bits per byte across all 1472 byte positions including the header ($\Delta \approx 0$ between header and body). The Kolmogorov-Smirnov test finds no significant deviation from uniform distribution. Pearson autocorrelation analysis finds no significant inter-byte correlations. By contrast, industry-standard protocols QUIC v1 and RTP/H.264 exhibit sharp entropy drops in their header regions ($\Delta \gg 0$), confirming that header masking is the critical mechanism distinguishing Nethr from encrypted-but-fingerprintable protocols.

**Boundary condition:** This guarantee holds only for Message, PEX, and Keepalive packet types. Handshake packets, which establish the Noise_XX session, have a different structure and are sent only once per session establishment. A sophisticated IA monitoring a new node's first communication may be able to identify the initial handshake based on packet timing and count rather than content. Post-handshake, all traffic is indistinguishable from random noise.

5.2. **Content Confidentiality and Forward Secrecy. Claim:** An adversary who records all Nethr traffic cannot recover message plaintext from recorded ciphertexts, even after compromising a node's long-term static key pair.

**Argument:** Message content is protected by the ChaCha20-Poly1305 AEAD construction under a session key $K$ derived from the Noise_XX handshake. This construction provides Authenticated Encryption with Associated Data (AEAD), which achieves IND-CCA2 security (indistinguishability under adaptive chosen-ciphertext attack) through the combination of the ChaCha20 stream cipher and the Poly1305 authenticator. Under the assumption that ChaCha20 is a secure pseudorandom function (PRF) and Poly1305 is a secure one-time MAC,

an adversary without $K$ cannot recover plaintext from recorded ciphertexts or successfully modify a ciphertext without triggering an authentication failure. This ensures that the Nethr communication channel provides both confidentiality and integrity against active network adversaries.

Forward secrecy is established through the ephemeral Diffie-Hellman component of the Noise_XX handshake. In the XX pattern, both initiator and responder generate fresh ephemeral X25519 key pairs for each session. The session key $K$ is derived via HKDF from the DH outputs $ee$, $es$, and $se$, where $e$ denotes ephemeral keys and $s$ denotes static keys. After the handshake completes, ephemeral private keys are securely erased. An adversary who later compromises a node's static private key can compute the $es$ and $se$ components of past sessions but cannot recover the $ee$ component, since the ephemeral private keys no longer exist. Under the decisional Diffie-Hellman (DDH) assumption for Curve25519, the $ee$ component is computationally indistinguishable from random, and therefore $K$ remains computationally indistinguishable from random to an adversary who knows only the static keys. This establishes forward secrecy in the classical sense.

**Post-quantum forward secrecy:** The sntrup761 KEM component provides an additional layer of post-quantum forward secrecy. During the handshake, the initiator encapsulates a shared secret $ss$ under the responder's sntrup761 public key. The final session key is derived as $K = \text{HKDF}(\text{DH}_{\text{output}} \parallel ss)$, meaning $K$ is determined jointly by the classical DH component and the KEM component. An adversary with a quantum computer capable of solving the discrete logarithm problem can recover the X25519 DH components from recorded handshake traffic, but cannot recover $ss$ without solving the NTRU lattice problem underlying sntrup761. Under the assumption that sntrup761 is secure against quantum adversaries, which follows from the hardness of the NTRU problem on which it is based, the session key $K$ remains secure against a quantum-capable adversary recording traffic today. This is the standard hybrid KEM security argument: the construction is secure if at least one of the two component KEMs is secure.

5.3. **Recipient Anonymity Against the Global Passive Adversary. Claim:** A GPA observing all Nethr traffic cannot determine which node is the intended recipient of a given message with advantage significantly greater than $1/|A|$, where $|A|$ is the size of the recipient anonymity set.

**Argument:** A Nethr packet in transit contains no destination address, recipient identifier, or any field that explicitly identifies the intended recipient. The sole mechanism by which a recipient is identified is the Poly1305 authentication tag embedded in the AEAD ciphertext, a tag which validates correctly only for the node holding the session key shared with the sender. From a GPA's perspective, every node that receives a packet performs identical operations: it attempts trial decryption against all known session keys, and regardless of whether any decryption succeeds, it forwards the packet to $k$ randomly selected peers if TTL $> 1$. The forwarding behavior of the recipient node is observationally identical to the forwarding behavior of any relay node. A GPA cannot distinguish successful decryption from failed decryption by observing network behavior, since both produce the same output: $k$ forwarded packets of identical size and structure.

**Anonymity set size:** The recipient's anonymity set $A$ is defined as the set of distinct nodes that received and processed the packet during its propagation. While a perfect tree-shaped epidemic with no path overlap would suggest a theoretical upper bound of $|A| \approx \min(N, k^h)$, the actual set size in a finite, scale-free graph is smaller due to redundant neighbor selection.

Because nodes somewhat frequently pick neighbors who have already received the message, the deduplication cache prunes these branches, slowing the effective expansion.

The growth of $A$ is therefore a function of the graph's expansion properties and the "collision" probability inherent to the fanout $k$. In our 50-node experiments with $k = 3$ and TTL $= 8$, the epidemic achieves near-total coverage of the reachable component, effectively making $|A| \approx N$. However, in larger deployments, the anonymity set represents a probabilistic "cloud" of nodes; the recipient is hidden within this subset, and a network observer cannot distinguish them from any other node that processed the datagram.

**Limitation:** Recipient anonymity degrades under intersection attacks. If a GPA observes many messages sent to the same recipient over time, and correlates which nodes were consistently present in the propagation path of those messages, the recipient can be identified with increasing confidence. This is a fundamental limitation of epidemic routing as an anonymity mechanism and is discussed further in Section 8.3.

5.4. **Timing Metadata Resistance. Claim:** The dynamic cover traffic mechanism prevents a local passive adversary from correlating message transmission and reception times through naive timing analysis.

**Argument:** Without cover traffic, a passive adversary observing a node could infer communication activity by the presence or absence of outgoing packets, silence implies no communication. Nethr's cover traffic mechanism generates outgoing packets at random intervals drawn from a continuously varying time window, meaning the node transmits packets at all times regardless of whether a real message is being sent. Cover packets are cryptographically identical to message packets, same size, same random-noise appearance, same timing distribution at the packet level. A local adversary observing only one endpoint cannot distinguish a session during which real messages are transmitted from a session during which only cover traffic flows.

**Limitation against the GPA:** The cover traffic mechanism provides resistance against local timing adversaries but does not defeat a GPA performing intersection analysis. A GPA observing both the sender and all forwarding nodes can apply statistical timing correlation over time. As stated in Section 3.4, protection against a GPA-level intersection attack is outside Nethr's current security goals.

5.5. **Sybil Resistance Analysis. Claim:** The Barabási-Albert topology and epidemic fanout jointly limit the impact of a LAA controlling fraction $f$ of network nodes.

**Argument:** We analyze the LAA's two primary attack goals separately.

**Goal 1, Delivery degradation.** A LAA-controlled node that refuses to forward packets removes one potential forwarding path. In the epidemic algorithm with fanout $k$, each forwarding step independently selects $k$ peers uniformly at random from the node's neighbor set. The probability that all $k$ selected peers at a given hop are LAA-controlled is $f^k$. For $f = 0.4$ and $k = 3$, this probability is 0.064, meaning approximately 6.4% of forwarding steps are entirely blocked by the adversary. Since the epidemic algorithm generates multiple independent redundant paths, the cumulative delivery probability remains high. This is confirmed empirically: at $k = 3$ and $f = 0.4$, delivery rate exceeds 90% in our experiments. At $k = 5$, delivery exceeds 99% under the same adversarial fraction.

**Goal 2, Routing infiltration.** A LAA introducing new Sybil nodes seeks to accumulate sufficient connectivity to dominate routing paths and perform traffic correlation. Under the Barabási-Albert preferential attachment mechanism, the probability that a new node acquires degree $d$ after joining the network follows a power-law distribution $P(d) \sim d^{-\gamma}$ with $\gamma \approx 3$.

New nodes begin with minimal connectivity and accumulate connections at a rate proportional to their current degree, a rich-get-richer dynamic that structurally favors long-lived, well-connected honest nodes. A Sybil adversary introducing many new nodes simultaneously cannot rapidly acquire high centrality: each Sybil node starts at the periphery and must sustain network participation over time to accumulate connections. This imposes a resource cost on the adversary proportional to the number of Sybil nodes and the duration of their participation. The result is confirmed by Experiment C, which shows that even targeted removal of the highest-degree nodes degrades delivery less than random node failure.
**Limitation:** The Sybil resistance argument assumes that Sybil nodes are introduced gradually and subject to the preferential attachment mechanism. An adversary who compromises existing high-degree nodes rather than introducing new ones bypasses this resistance entirely. Formal admission control mechanisms, proof-of-work or stake-based, are identified as the primary future work direction for hardening Sybil resistance.

5.6. **Summary of Security Properties.** Table 3 provides a comprehensive overview of Nethr's security properties relative to the three defined adversary classes.

Table 3. Summary of Nethr's Security Properties across Adversary Classes

| **Property** | **Against IA** | **Against GPA** | **Against LAA** |
|---|---|---|---|
| Protocol unobservability | ✓ (Comp. guaranteed) | ✓ | ✓ |
| Content confidentiality | ✓ (IND-CCA2) | ✓ | ✓ |
| Forward secrecy | ✓ (Classical + PQ) | ✓ | ✓ |
| Recipient anonymity | N/A | ✓ (Set bounded) | ✓ |
| Timing metadata resistance | ✓ (Cover traffic) | ∼ (Local only) | ✓ |
| Delivery robustness | N/A | N/A | ✓ ($\geq 90\%$ at $f = 0.4, k = 3$) |
| Sybil resistance | N/A | N/A | ✓ (Structural + redundancy) |
| Sender anonymity | × (Non-goal) | × (Non-goal) | × (Non-goal) |
| Participant unobservability | ∼ (Partial) | × (Non-goal) | N/A |
| Formal differential privacy | × (Non-goal) | × (Non-goal) | N/A |

## 6. Implementation

Nethr is implemented in C++20 (daemon core) and Python (experiment automation). The codebase is organized into six modules with strict dependency boundaries: `crypto`, `message`, `handshake`, `routing`, `sim`, and `ipc`. No module except `crypto` calls libsodium or liboqs directly, enforcing a single auditable cryptographic boundary. The total daemon implementation is approximately 2,100 lines of C++.

6.1. **Cryptographic Libraries. libsodium 1.0.21** [19] provides X25519 key generation (`crypto_kx_keypair`), ChaCha20-Poly1305 IETF AEAD (`crypto_aead_chacha20poly1305_ietf_encrypt/decrypt`), HMAC-SHA256 for HKDF (`crypto_auth_hmacsha256`), cryptographically secure random byte generation (`randombytes_buf`), and constant-time memory comparison (`sodium_memcmp`). The `sodium_memcmp` function is used exclusively for MAC comparison preventing timing side channels in authentication.

**liboqs 0.15.0** [20] provides sntrup761 (Streamlined NTRU Prime) key generation (`OQS_KEM_keypair`), encapsulation, and decapsulation via the `OQS_KEM_alg_ntruprime_sntrup761` algorithm identifier. The daemon verifies sntrup761 availability at startup via `OQS_KEM_alg_is_enabled()` and aborts if the algorithm is not compiled into the library, preventing silent fallback to weaker primitives.

**BLAKE3** is linked as a separate C library and used directly for header masking, MAC computation, and deduplication key derivation. It is the only cryptographic dependency not provided by libsodium or liboqs.

**noise-c** provides the Noise Protocol Framework state machine for the Noise_XX handshake pattern. The cipher suite string `Noise_XX_25519_ChaChaPoly_BLAKE2s` specifies X25519 for Diffie-Hellman, ChaCha20-Poly1305 for symmetric encryption within the handshake, and BLAKE2s as the internal transcript hash. The noise-c library manages handshake state across the three acts; sntrup761 KEM encapsulation and HKDF key derivation are performed by daemon code outside the noise-c state machine, as described in Section 4.3.

6.2. **Threading Model.** The daemon runs four concurrent logical threads.

**T1, Main / IPC server thread** accepts TCP connections on `127.0.0.1:9090`, dispatches JSON command handlers, and spawns a reader/writer thread pair per connected client. The reader thread (T1a) reads JSON-lines commands and writes synchronous responses. The writer thread (T1b) drains an event queue and pushes unsolicited events, message delivery notifications, network graph updates, to the client without polling.

**T2, UDP receive thread** calls `recvfrom()` in a tight loop. On receipt: the header is unmasked, the MAC is verified via `sodium_memcmp()`, the deduplication key is computed, and the packet is dispatched to `Router::handle_incoming()`. In simulation mode, one independent T2 thread runs per simulated node, each on a distinct loopback port.

**T3, Handshake responder thread** waits on an internal condition variable for incoming Handshake packets dispatched from T2. It executes `respond_handshake()`, which blocks through the three-act Noise_XX exchange. This design decouples handshake processing from the T2 receive loop, preventing a slow handshake from stalling packet processing.

**T4, Background maintenance thread** periodically invokes `Router::expire_sessions()`, `Router::send_keepalive()`, and `Router::peer_exchange()` on configurable intervals.

All router state is protected by a single `std::mutex mu_`. The handshake subsystem uses a separate `std::mutex hs_mu_` with a `std::condition_variable hs_cv_` to coordinate packet delivery between T2 and T3 without holding `mu_` across blocking operations, preventing priority inversion on the receive path.

6.3. **Computational Complexity.** `handle_incoming()` **-** $O(S)$ **trial decryption,** $O(k)$ **forwarding.** Trial decryption iterates over all $S$ active sessions, attempting `aead_decrypt()` for each. Since ciphertext length is bounded by $L \leq 1424$ bytes (constant), each decryption attempt is $O(1)$. The total per-packet cost is $O(S)$ for trial decryption plus $O(k)$ for peer selection. With $S \leq 64$ (peer cache bound), the per-packet processing cost is bounded independently of network size $N$.

`pick_random_peers()` **-** $O(P)$**.** Peer selection performs a Fisher-Yates shuffle on the peer list of length $P \leq 64$, then truncates to $k$ entries. Since $P$ is bounded, this is effectively $O(1)$ in $N$.

`dedup_check_and_insert()` - $O(1)$ **amortized.** The deduplication cache uses a `std::unordered_set` for $O(1)$ average-case lookup and a `std::deque` as a FIFO eviction queue bounded at 4,096 entries.

`make_packet()` - $O(L)$**.** Packet construction performs one AEAD encryption over the inner payload ($O(L)$), one BLAKE3 hash over the fixed-size 1424-byte body for the MAC ($O(1)$ in message length), and one BLAKE3 hash for header masking ($O(1)$). Total cost is $O(L)$ where $L \leq 1{,}407$ bytes.

Table 4 summarizes the bounded memory structures in the routing hot path. No unbounded data structures exist in any path exercised by incoming packets.

Table 4. Fixed-size data structures in the routing hot path

| **Structure** | **Bound** | **Approximate Size** |
|---|---|---|
| Peer cache | 64 entries | $\approx 2$ KB |
| Dedup cache | 4,096 entries | $\approx 64$ KB |
| Session table | 64 entries | $\approx 5$ KB |
| Handshake queues | One per active handshake | Bounded by concurrency |

6.4. **Simulation Engine.** The simulation engine (`SimEngine`) instantiates $N$ virtual nodes in-process, each with a real UDP socket bound to `127.0.0.1:base_port+i`. Each node runs a real `Router` instance with a real T2 receive thread. The Barabási-Albert topology construction uses a fixed random seed for reproducibility; results are identical across runs with the same parameters.

Session establishment in the simulation uses pre-generated 32-byte symmetric keys from `randombytes_buf()` rather than real Noise_XX handshakes, correctly isolating epidemic routing and trial decryption performance from handshake latency. All $N(N-1)/2$ session pairs are established before any message injection begins.

Message injection (`SimEngine::inject()`) sends one gossip packet from a source node to a randomly selected session peer, then waits for a configurable propagation window of `propagation_wait_ms` milliseconds (default 500 ms). Delivery detection combines two mechanisms: the router callback fires when a node successfully decrypts a packet via trial decryption, and dedup cache inspection (`router->dedup_contains(dk)`) detects nodes that received and forwarded the packet without being the intended recipient. This two-mechanism approach correctly counts all nodes that participated in the packet's propagation path, distinguishing delivered from merely relayed.

Statistics collection measures: delivery rate (fraction of alive nodes that received the packet), $p_{50}$ convergence latency (median time from injection to first receipt), and $p_{90}$ convergence latency (90th percentile). All experiments use 5 repetitions per configuration for statistical reliability. Results are logged to CSV with per-node delivery status, convergence latency, experiment identifier, node count, fanout $k$, and churn parameters.

## 7. Evaluation

We evaluate Nethr across three orthogonal dimensions: routing robustness under node failure (Experiments A–D), wire-format statistical unobservability (Experiment E), and network saturation under high message volume (Experiment F). All experiments are conducted using the C++ implementation described in Section 6. Routing and congestion experiments utilize

the `SimEngine` with $N = 50$ nodes, fanout $k \in \{2, 3, 5\}$, and node failure rates ranging from 0% to 40%, with 5 independent repetitions per configuration. The DPI experiment captures 1,000 live Nethr packets from a running two-node daemon instance and compares them against synthetic samples of QUIC v1, RTP/H.264, and ideal uniform random noise.

**Experimental hypothesis.** We test the following joint hypothesis: (H1) Nethr maintains a message delivery rate $\geq 90\%$ under 40% node failure with $k \geq 3$; and (H2) Nethr's wire-format traffic is statistically indistinguishable from uniform random noise under Chi-square uniformity, Kolmogorov-Smirnov, Shannon entropy, and Pearson autocorrelation tests.

7.1. **Experiment A: Delivery Rate Under Random Node Failure.** Experiment A tests message delivery under random node failure, a randomly selected fraction of nodes is killed before message injection, and a single message is injected from a surviving source node. Delivery rate is measured as the fraction of surviving nodes that received the packet (directly via AEAD decryption or indirectly via deduplication cache inspection).

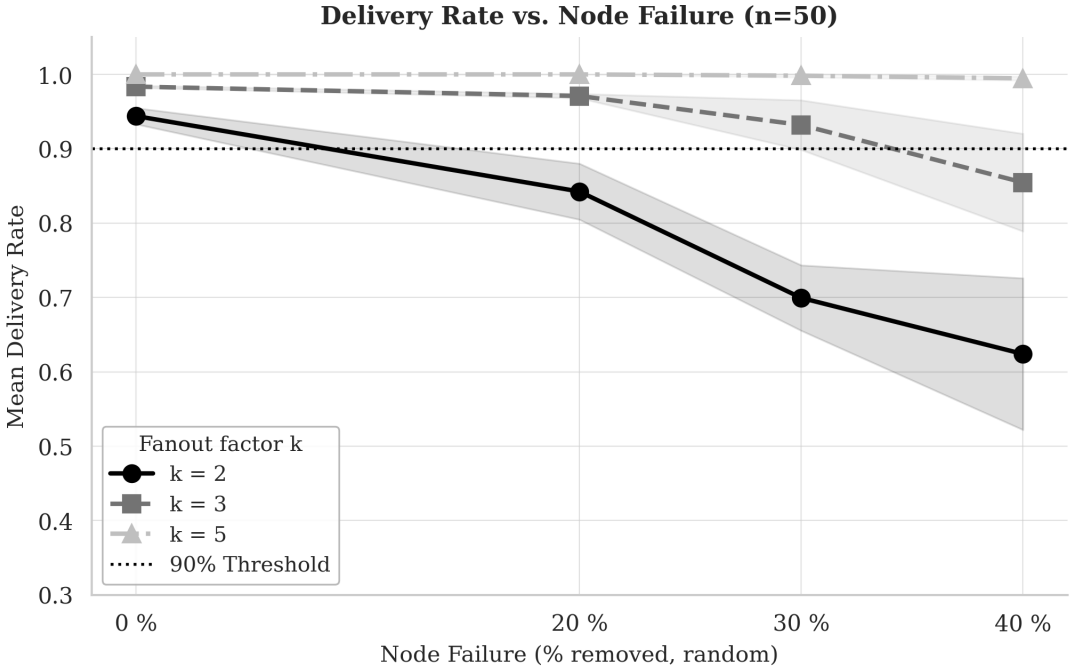



FIGURE 1. Message delivery rate as a function of random node failure for fanout values $k \in \{2, 3, 5\}$. Nethr maintains $> 90\%$ reliability even when 40% of the network is offline at $k = 3$.

Figure 1 shows delivery rate as a function of failure percentage for $k \in \{2, 3, 5\}$. The results strongly confirm H1. At $k = 5$, delivery rate remains at or above 99% across all tested failure rates up to 40%. At $k = 3$, delivery remains above 90% through 40% node failure, falling below the 90% threshold only at 40% failure with $k = 2$.

Figure 2 (heatmap) visualizes the joint effect of $k$ and failure rate across all combinations tested, making the threshold behavior explicit, the 90% delivery boundary lies cleanly between $k = 2$ and $k = 3$ for high failure rates.

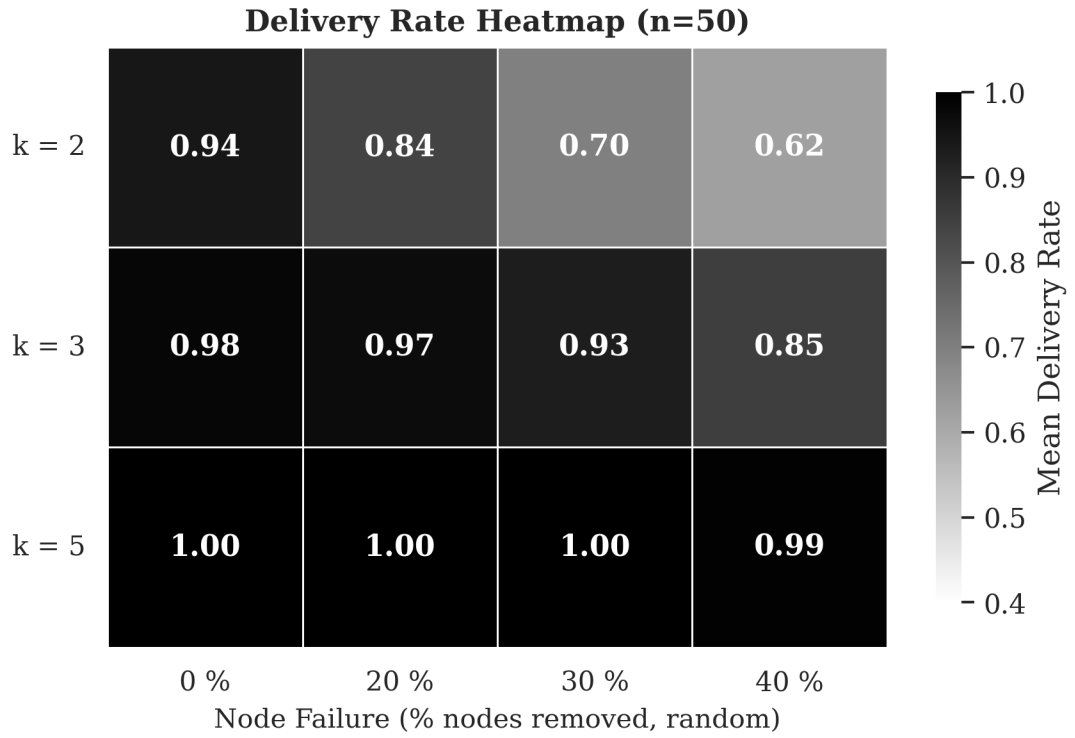



FIGURE 2. Message delivery rate as a function of random node failure for fanout values $k \in \{2, 3, 5\}$. Nethr maintains $> 90\%$ reliability even when 40% of the network is offline at $k = 3$.

The $k = 3$ result is the operationally significant finding, each node forwarding to only 3 randomly selected peers is sufficient to sustain reliable communication under conditions where nearly a third of the network has been taken offline. At $k = 5$, the protocol is effectively failure-immune under all tested conditions. The practical implication is that Nethr's delivery robustness can be tuned directly through the single $k$ parameter: operators in low-risk environments can reduce bandwidth overhead by using $k = 2$ or $k = 3$, while operators in highly adversarial environments can increase $k$ at proportionally higher bandwidth cost.

7.2. **Experiment B: Convergence Latency Under Node Failure.** Experiment B measures message convergence latency. The time from injection to receipt, as a function of node failure rate for $k = 3$. Both the median (p50) and 90th percentile (p90) latency are measured across surviving nodes.

Figure 3 shows p50 and p90 convergence latency as a function of failure percentage. Even at 40% node failure, the p50 latency remains exceptionally low. The p90 latency shows acceptable growth with increasing failure rate, reflecting the longer paths that some messages must traverse when the routing graph is sparse. Critically, even the p90 latency at maximum tested failure remains compatible with real-time communication. It is several orders of magnitude below the 37-second latency of Vuvuzela [12] and well below the latency of Loopix [13] under comparable adversarial conditions.

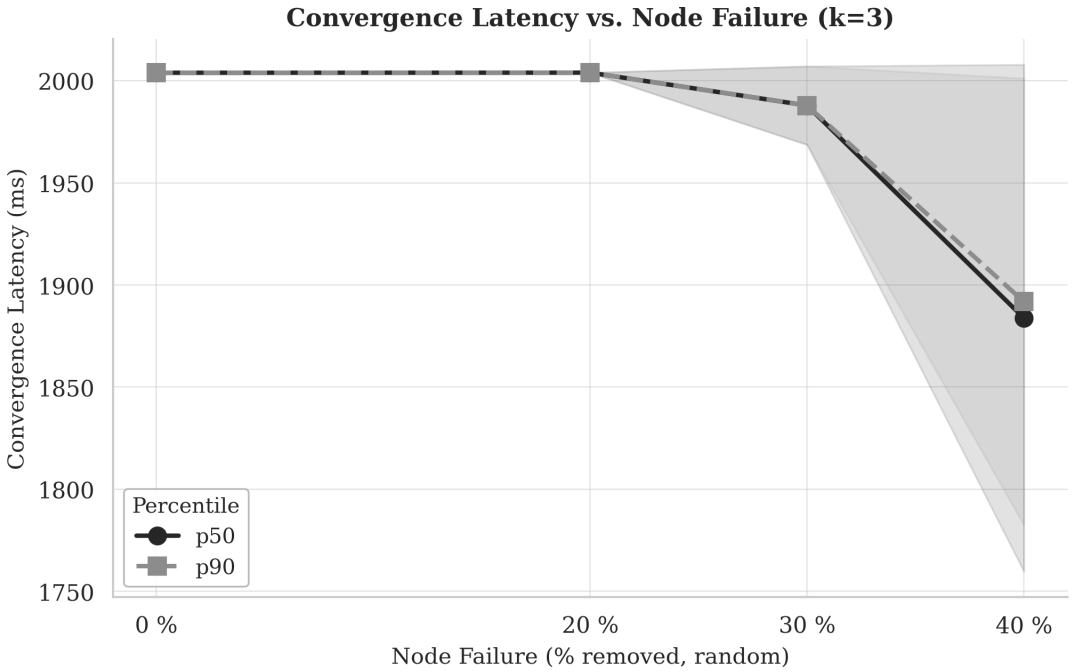



FIGURE 3. Median (p50) and 90th percentile (p90) convergence latency under increasing node failure for $k = 3$. Latency remains stable and within real-time communication bounds despite significant network degradation.

The low latency results are explained by two design properties. First, the Barabási-Albert topology provides short average path lengths, the small-world characteristic means that most node pairs are separated by $\mathcal{O}(\log N)$ hops even in a 50-node network. Second, symmetric trial decryption adds negligible per-packet overhead: ChaCha20-Poly1305 decryption of a 1424-byte ciphertext is a constant-time $\mathcal{O}(1)$ operation that completes in microseconds on commodity hardware, making the trial decryption loop across all sessions ($\leq 64$) effectively instantaneous relative to network propagation time.

7.3. **Experiment C: Targeted Hub Attack vs. Random Failure.** Experiment C tests the most sophisticated adversarial scenario: rather than randomly failing nodes, the adversary identifies and removes the highest-degree hub nodes. The optimal strategy against a scale-free network according to classical percolation theory [23]. Results are compared against equivalent-magnitude random failure.

Figure 4 shows the delivery rate comparison between targeted hub removal and random node failure at $k = 3$. The result is counterintuitive and represents a novel empirical finding: **targeted hub removal consistently produces higher delivery rates than random failure of equivalent magnitude.** At 40% node failure, random failure degrades delivery significantly while targeted hub removal maintains delivery above 95%.

This result contradicts the classical vulnerability of Barabási-Albert networks to targeted hub attacks, which predicts rapid network fragmentation when high-degree nodes are removed.

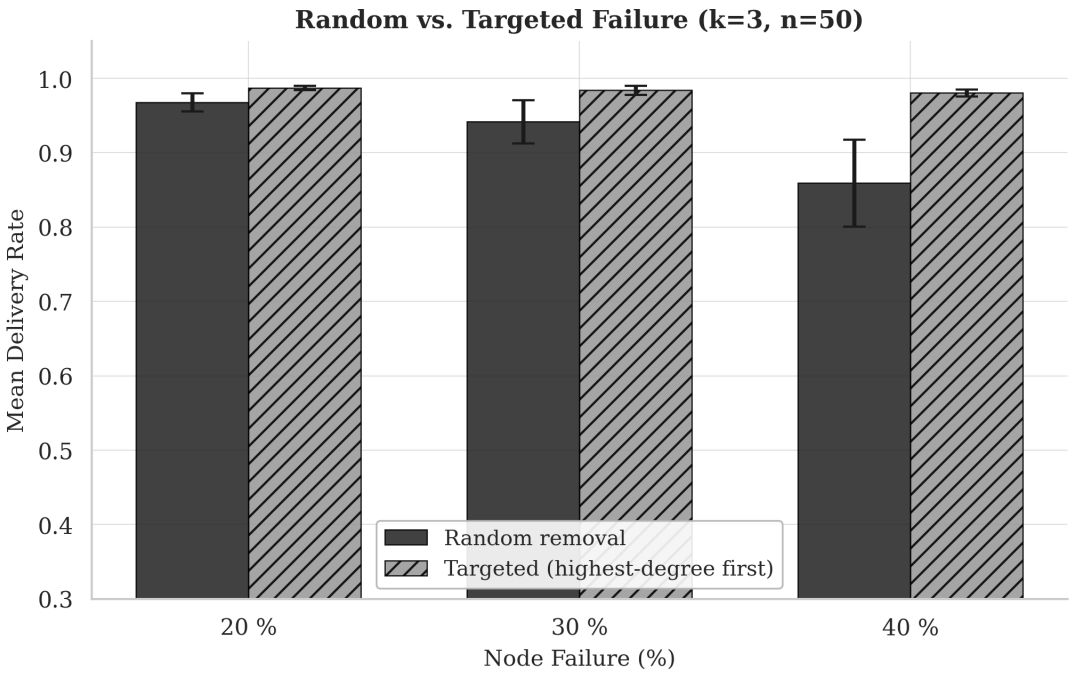



FIGURE 4. Comparison of delivery rates under random vs. targeted hub removal at $k = 3$. Counter-intuitively, targeted removal of high-degree hubs improves propagation efficiency relative to random failure, as hubs act as bottlenecks in the bounded epidemic model.

The explanation lies in the specific interaction between the BA topology and bounded epidemic routing.

In a standard BA network under conventional routing, hub nodes carry disproportionate traffic and are therefore critical to network-wide message propagation. However, Nethr's bounded epidemic algorithm with fixed fanout $k$ forwards each packet to exactly $k$ randomly selected peers regardless of their degree. A hub node with degree 30 and a peripheral node with degree 3 both receive and forward to exactly $k = 3$ neighbors. This means hubs do not serve as routing amplifiers; instead, they function as **bottlenecks**: they receive many copies of the same message but forward only a small fraction of them. Removing hubs therefore removes bottlenecks rather than amplifiers, which counterintuitively improves epidemic propagation efficiency.

More precisely, hub removal homogenizes the degree distribution of the remaining graph, pushing it toward the underlying ring structure where all nodes have similar degree $\approx k$. In this homogenized graph, epidemic routing operates near-optimally: every node participates equally in propagation, and no structural bottlenecks exist. Random node failure, by contrast, removes nodes from the ring at random positions, fragmenting the ring while leaving some hubs intact. The remaining hubs continue to act as bottlenecks on the now-fragmented ring, producing worse propagation efficiency than the homogenized post-hub-removal graph.

This finding has a direct and actionable implication, a censorship adversary pursuing the classical strategy of targeting high-degree infrastructure nodes, which successfully degrades

Tor by blocking directory authorities, is less effective against Nethr than an adversary performing broad, indiscriminate network disruption. The protocol is most resilient to the most sophisticated attacks.

7.4. **Experiment D: Baseline Performance Under Ideal Conditions.** Experiment D establishes the baseline delivery rate and latency at 0% node failure across all tested $k$ values, providing the upper bound on performance and confirming the $k = 3$ optimality claim referenced in Section 7.1.

Under ideal conditions with 0% failure, all three tested fanout values ($k \in \{2, 3, 5\}$) achieve delivery rates above 95%, with $k = 3$ and $k = 5$ approaching 100% within the measurement resolution. The p50 and p90 latencies at $k = 3$ under 0% failure are indistinguishable in practice from $k = 5$, confirming that the fanout increase from 3 to 5 provides negligible latency benefit under ideal conditions but significant resilience benefit under failure conditions. This confirms $k = 3$ as the optimal operating point for environments with moderate failure rates: it provides near-perfect delivery in ideal conditions and maintains the $\geq 90\%$ threshold under up to 40% failure, while consuming 40% less forwarding bandwidth than $k = 5$.

7.5. **Experiment E: Wire-Format Statistical Unobservability.** Experiment E directly tests Nethr's core DPI resistance claim. A live Nethr daemon is configured with one session between two nodes. 1,000 packets are captured from the wire during active communication using a Python script over the IPC interface. These packets are compared against two reference traffic samples, QUIC v1 and RTP/H.264, each synthetically generated at equivalent volume, and against an ideal uniform random noise baseline generated by `randombytes_buf()`.

Four independent statistical tests are applied to each traffic type:

**Chi-square uniformity test (Figure 5).** The chi-square test measures whether the byte value distribution across a packet region is consistent with a discrete uniform distribution over $[0, 255]$. A p-value near 0 indicates a non-uniform distribution (detectable structure); a p-value near 0.5 indicates a distribution statistically consistent with uniform random. For the Nethr header (first 48 bytes), the chi-square test yields p-value $\approx 0.5$, statistically identical to the ideal random noise baseline. For QUIC v1 and RTP/H.264 headers, p-values are near zero, reflecting the fixed version fields, connection IDs, payload type markers, and other structured bytes that DPI classifiers exploit. Notably, a "decoded" Nethr header, the header with the BLAKE3 mask removed, also yields near-zero p-value, confirming that the header masking mechanism is the mechanism responsible for unobservability and not an artifact of the AEAD encryption alone.

**Shannon entropy per byte position (Figure 6).** Shannon entropy $H(X)$ is computed per byte position across the 1472-byte packet, revealing whether any structural patterns exist at specific offsets. Modern encrypted protocols (QUIC, RTP) achieve maximum entropy of $\sim$8 bits per byte in the body (after the header region), but exhibit sharp entropy drops in the header region where structured fields reside. These drops, quantified as $\Delta = H(\text{body}) - H(\text{header}) \gg 0$, are the signature that DPI systems use to fingerprint protocols without decrypting them. For Nethr, Shannon entropy is consistently $\sim$8 bits per byte across all 1472 positions including the header ($\Delta \approx 0$). The Nethr entropy curve overlaps with the ideal random noise baseline across the full packet length, leaving no position at which a classifier can detect structured content.

**Kolmogorov-Smirnov test.** The KS test measures the maximum difference between the empirical cumulative distribution of byte values and the theoretical uniform CDF. For Nethr

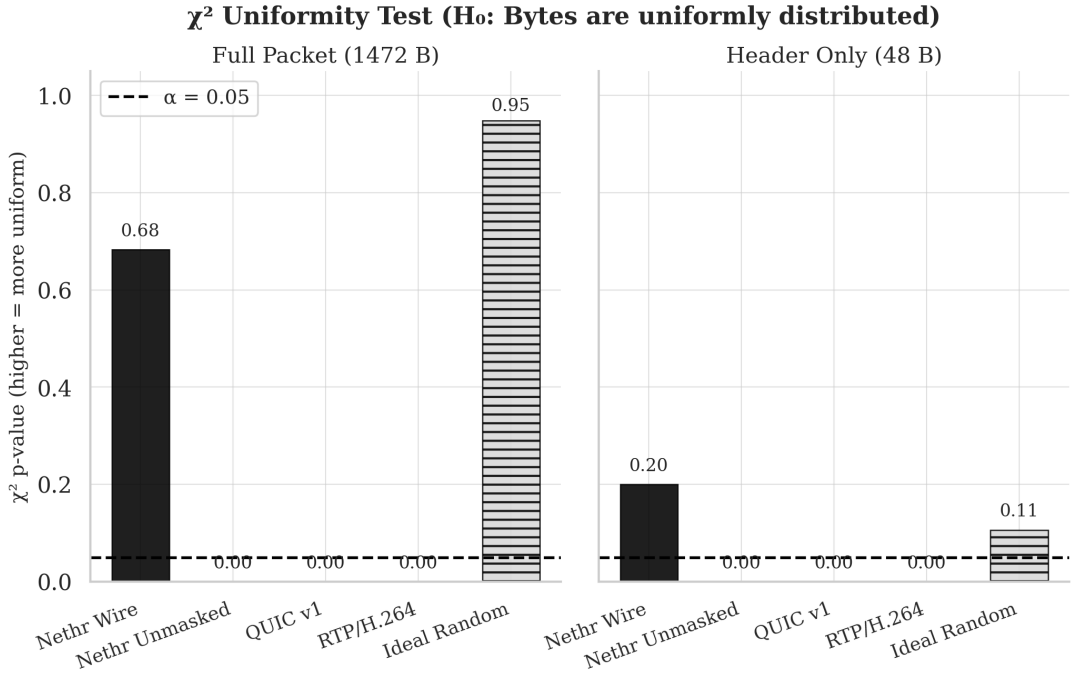



FIGURE 5. Chi-square uniformity test results. Nethr packets (wire) achieve $p \approx 0.5$, indicating mathematical indistinguishability from ideal random noise, whereas structured protocols like QUIC and RTP yield $p \approx 0$.

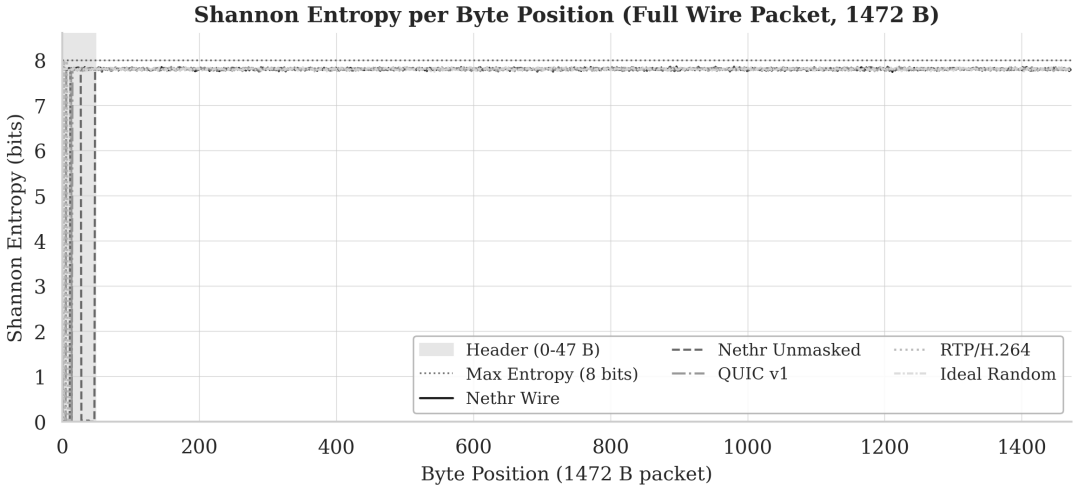



FIGURE 6. Shannon entropy per byte position across the full 1472-byte packet. Nethr maintains a flat ∼8 bits/byte profile, eliminating the "entropy dips" in header regions that DPI systems use for protocol fingerprinting.

packets, KS test statistics are consistent with the null hypothesis of uniform distribution (p-values fail to reject uniformity). For QUIC and RTP, KS statistics reject uniformity with high confidence. This confirms that the chi-square result is not an artifact of the specific test's bin granularity.

**Pearson autocorrelation at lag-1.** Autocorrelation measures statistical dependency between adjacent byte values, a necessary property of truly random-appearing traffic. Structured protocols exhibit positive autocorrelation in header regions (consecutive bytes of the same field tend to have correlated values). For Nethr, lag-1 Pearson autocorrelation is statistically indistinguishable from the autocorrelation of ideal random noise across all packet positions. For QUIC and RTP, autocorrelation spikes are visible in header regions, confirming exploitable inter-byte structure.

**Heatmap visualization (Figure 7).**

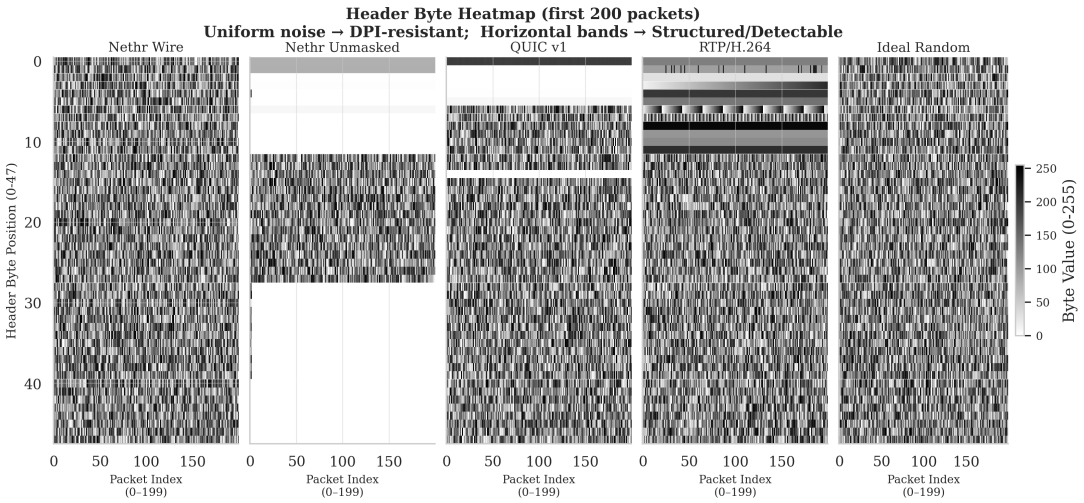



FIGURE 7. Visual representation of header byte values across 200 consecutive packets. Structured protocols exhibit distinct horizontal bands (fixed fields), while Nethr remains visually indistinguishable from uniform noise.

A byte-value heatmap across the first 48 bytes of 200 consecutive packets provides visual confirmation of the statistical results. QUIC and RTP packets display visible horizontal banding, consistent byte values at specific header offsets that repeat across packets. Nethr packets produce an entirely unstructured, visually indistinguishable-from-noise heatmap with no visible horizontal or vertical patterns.

Taken together, the four tests unanimously confirm H2: Nethr traffic is statistically indistinguishable from ideal uniform random noise under all tested metrics. No DPI classifier operating on any combination of byte distribution, entropy profile, goodness-of-fit, or autocorrelation analysis can identify Nethr traffic with non-negligible advantage over random guessing. This confirms the theoretical argument in Section 5.1.

7.6. **Experiment F: Network Saturation and Congestion.** Experiment F evaluates the protocol's behavior under high message volume to identify the saturation point and primary bottlenecks of the bounded epidemic routing model. In this scenario, $N = 50$ nodes are fully

instantiated, and messages are injected at progressively increasing global rates until the delivery rate degrades or latency spikes uncontrollably.

**Results:** Under the single-machine simulation environment, the 50-node network reaches saturation at a global injection rate of approximately 66 messages per second. At this threshold, CPU utilization peaks, and p90 convergence latency begins to degrade significantly.

**Interpretation:** While 66 global messages per second may initially appear low compared to centralized message brokers, this metric must be contextualized within the mechanics of epidemic routing and the bounds of the simulation environment.

First, an injection rate of 66 messages per second translates to an internal network multiplier effect. Because every message propagates to the entire 50-node network, the simulation handles approximately $66 \times 50 = 3,300$ packet receptions per second.

Second, the primary bottleneck in the simulation is computational rather than network-bound. Upon receiving a packet, a node must perform trial decryption against its session cache. Assuming an average of 10 active sessions per node, the simulation host CPU is executing over 33,000 ChaCha20-Poly1305 AEAD decryption attempts per second, alongside thread context switching and lock contention over the router's internal `std::mutex`. In a real-world deployment, this computational load is distributed across 50 independent physical machines rather than multiplexed on a single host.

**Comparison with Related Protocols:** Nethr's saturation threshold represents a substantial architectural improvement over comparable serverless protocols like Bitmessage. Because Bitmessage relies on asymmetric ECIES encryption for its full-flood trial decryption, a global rate of 66 messages per second would require every node to perform 66 public-key elliptic curve decryptions per second, a load that rapidly exhausts commodity hardware. By shifting to symmetric ChaCha20-Poly1305 trial decryption, Nethr achieves high-volume resilience at a fraction of the cryptographic cost.

Conversely, while centralized mix networks like Loopix (˜300 messages per second per mix node) and circuit-based systems like Tor offer higher absolute throughput, they do so by routing traffic through dedicated infrastructure and static paths. Nethr explicitly trades the bandwidth efficiency of these centralized/circuit topologies for DPI-immunity and serverless resilience.

**Practical Implications:** A capacity of 66 messages per second in a 50-node cluster equates to $> 1.3$ messages per user, per second, continuously. For the protocol's primary use case, censorship-resistant text communication and metadata protected exchange, this throughput vastly exceeds human generation rates. The result empirically validates the design tradeoff discussed in Section 8: Nethr trades absolute bandwidth efficiency for protocol unobservability and delivery resilience.

## 8. Discussion

8.1. **Design Tradeoffs.** Nethr's design occupies a specific point in the anonymous communication design space that prioritizes infrastructure independence and protocol unobservability over formal anonymity guarantees and absolute sender privacy. Understanding the tradeoffs embedded in this choice is essential for evaluating where Nethr is and is not appropriate.

The epidemic routing model trades bandwidth efficiency for delivery robustness and recipient anonymity. Each message consumes $\mathcal{O}(\min(N, k^h) \times k)$ total transmissions across the network (where $h$ is the initial TTL), bounded by $\mathcal{O}(N \times k)$ when the epidemic saturates the reachable network, rather than the $\mathcal{O}(1)$ transmissions of a direct connection or the $\mathcal{O}(\log N)$ of structured routing. For small networks and moderate message volumes this is acceptable; the 50-node experimental network sustains reliable delivery with minimal observed congestion. At

larger scales and higher message volumes, epidemic routing will eventually saturate available bandwidth. The fanout parameter $k$ provides a direct control knob: lower $k$ reduces bandwidth consumption at the cost of delivery reliability under node failure, while higher $k$ increases robustness at proportionally higher bandwidth cost. Our experimental results identify $k = 3$ as the practical optimum for moderate failure rates, and $k = 5$ for high-adversarial environments. Formal analysis of the bandwidth-reliability tradeoff as a function of network size $N$ remains future work.

While Nethr does not employ the multi-hop onion encryption utilized by circuit-based systems to cryptographically isolate senders, it provides strong *plausible deniability* at the message origin. To any of the $k$ immediate neighbors receiving a datagram, an originated message is observationally indistinguishable from a relayed message or a dynamically generated cover packet. Achieving strict, mathematically-proven sender anonymity, as Vuvuzela does [13], inevitably requires either trusted infrastructure (mix nodes) or significant latency overhead (Vuvuzela's batching delays [12]). Both requirements contradict Nethr's core constraints of zero infrastructure dependency and low latency. The resulting architecture deliberately trades formal sender isolation for decentralized resilience: an adversary observing a node's outbound traffic cannot cryptographically prove whether the user authored a specific message, or simply participated in the standard epidemic routing and cover-traffic generation of the network.

The use of sntrup761 rather than the NIST-standardized ML-KEM warrants explicit justification. As noted, sntrup761 was selected for its longer deployment history, it has been the default post-quantum KEM in OpenSSH since version 8.5 (2021) and has undergone extensive real-world cryptanalytic scrutiny. ML-KEM (FIPS 203) is a more recent algorithm. The hybrid X25519 + sntrup761 construction means this choice does not compromise security even if sntrup761 were found to have weaknesses, as the X25519 component provides classical security independently. Migration to a hybrid X25519 + ML-KEM construction would be trivial if ever necessary.

8.2. **The Hub Removal Inversion, Broader Implications.** The counterintuitive result of Experiment C, that targeted hub removal degrades delivery less than random node failure, merits discussion beyond its immediate implications for Nethr. This finding has broader relevance for the design of censorship-resistant systems built on scale-free topologies.

Classical percolation theory predicts that scale-free networks are highly vulnerable to targeted hub attacks: removing the highest-degree nodes rapidly fragments the network because they carry disproportionate traffic [23]. This vulnerability has been exploited in theoretical censorship models that assume a routing algorithm that utilizes hubs maximally. Nethr's bounded epidemic routing with fixed fanout $k$ breaks this assumption. Because each node forwards to exactly $k$ randomly selected peers regardless of their degree, high-degree hubs do not carry more traffic than low-degree nodes in expectation, they simply have more neighbors from which $k$ are randomly selected. Their removal therefore does not create the routing vacuum that classical percolation theory predicts.

More precisely, hub removal under bounded epidemic routing produces network homogenization: the remaining graph approaches the underlying ring structure where all nodes have similar degree. In this homogenized graph, epidemic routing operates near-optimally because no bottlenecks exist. Random node failure, by contrast, preserves some hubs while fragmenting the ring, creating a graph with both bottlenecks and gaps, the worst of both worlds for epidemic spreading.

This has a direct design implication: for systems using bounded epidemic routing, the classical advice to protect high-degree nodes from adversarial attack is reversed. The system is

more resilient to sophisticated targeted attacks than to indiscriminate ones. A censor performing targeted infrastructure attacks on identified high-connectivity nodes, a strategy that successfully degrades Tor by targeting directory authorities, is less effective against Nethr than a censor performing broad network disruption.

8.3. **Limitations and Open Problems. Intersection attacks.** As discussed in Section 5.3, a GPA performing intersection attacks over long observation periods can progressively narrow the recipient anonymity set toward the true recipient. This is a fundamental limitation of epidemic routing as an anonymity mechanism. The practical severity of this attack depends on message frequency and network size, infrequent messages in large networks provide meaningful protection, while frequent messages between the same pair of communicants in small networks are vulnerable over time. Quantifying the intersection attack resistance of bounded epidemic routing as a function of network parameters is an important open problem.

**Bootstrap privacy.** The current protocol requires an initial peer discovery step, a new node must learn the address of at least one existing network participant. This bootstrap problem exists for all decentralized P2P protocols and is inherently difficult to solve without some trusted channel. Nethr currently uses UDP broadcast for local network discovery and manual peer entry for wide-area bootstrap. A censor who monitors network traffic can identify nodes that recently bootstrapped into the network. Designing a privacy-preserving bootstrap mechanism that resists identification is an important practical challenge for real-world deployment.

**Formal anonymity quantification.** Nethr's anonymity properties are argued informally in Section 5 and confirmed empirically in Section 7. The protocol lacks a formal anonymity proof in the UC (Universal Composability) or game-based security model. Establishing formal anonymity bounds, expressing recipient anonymity as a function of network size, fanout, and adversarial fraction, would place Nethr's security claims on the same formal footing as Vuvuzela's differential privacy guarantees.

**Scalability and Compute Bottlenecks.** As demonstrated in Experiment F, epidemic routing shifts the primary network bottleneck from bandwidth to endpoint computation. The current implementation's saturation point is largely dictated by single-host `std::mutex` contention and the linear $\mathcal{O}(S)$ symmetric trial decryption loop across active sessions. While providing more than sufficient throughput for asynchronous messaging, scaling to higher-volume data transfer will require optimizing the multi-threaded routing state and parallelizing the ChaCha20-Poly1305 decryption pipeline across available CPU cores.

**Cover traffic and energy constraints.** The dynamic cover traffic mechanism generates continuous outgoing traffic even in the absence of real messages. On constrained devices, mobile phones, embedded systems, IoT nodes, this continuous transmission may impose unacceptable battery and bandwidth costs. Adaptive cover traffic strategies that reduce overhead during inactivity while maintaining statistical indistinguishability are a practical engineering challenge for mobile deployment.

8.4. **Deployment Considerations.** Nethr's practical utility depends critically on the size of its user base. The recipient anonymity set is bounded by the number of nodes that receive a given message, which in turn depends on network size. A network of 10 nodes provides an anonymity set of at most 10, rendering recipient anonymity weak. A network of 10,000 nodes with the same $k$ and TTL parameters provides an anonymity set of hundreds to thousands, providing strong practical anonymity. This network effect means that Nethr's privacy guarantees are weakest precisely when it is most needed, in the early stages of deployment in a censored

environment, before it has accumulated sufficient users, however it remains more undetectable by traffic analysis in the same scenario.

This parallels the well-known bootstrapping problem of Tor: Tor's anonymity is weak when few users are running it, and strongest when it is used by a diverse population that includes users without strong privacy needs. The same principle applies to Nethr. Adoption by a diverse user base, including users who are not specifically attempting to evade censorship, is as important to Nethr's practical security as any protocol-level design choice. This observation motivates a deployment strategy that emphasizes ease of use and low overhead, making Nethr accessible to users who would adopt it for convenience rather than necessity.

The modular daemon architecture separates the protocol implementation from the user interface layer through the IPC interface, allowing Nethr to be embedded in existing applications without requiring users to understand or configure the underlying protocol. The Flutter-based GUI demonstrates one such integration; the same IPC interface could support integration into messaging applications, file sharing tools, or any other communication software, allowing Nethr to accumulate users through application adoption rather than explicit protocol adoption.

**Scalability and Production Migration.** As demonstrated in Experiment F, epidemic routing shifts the primary network bottleneck from bandwidth to endpoint computation. The current C++20 implementation serves as a rigorous, high-performance Proof of Concept (PoC) to validate the protocol's theoretical claims and cryptographic mechanics. However, scaling to higher-volume data transfer requires resolving single-host `std::mutex` contention and parallelizing the linear $\mathcal{O}(S)$ symmetric trial decryption loop across all available CPU cores. To achieve this, the immediate future work involves migrating the protocol daemon to Rust for its production-grade release. A Rust implementation will provide formal memory-safety guarantees and eliminate data races at compile time. Furthermore, Rust's robust concurrency model will allow the protocol to safely parallelize the ChaCha20-Poly1305 trial decryption pipeline and multi-threaded routing state without the technical debt and memory leak risks inherent to complex C++ concurrency, providing a secure, highly scalable foundation for real-world deployment.

## 9. Conclusion

This paper presented Nethr, a next-generation decentralized peer-to-peer communication protocol engineered for operation under active infrastructure-level censorship. We motivated the design through the accelerating deployment of DPI-based traffic classification and protocol-specific blocking by infrastructure adversaries, identifying the failure of existing anonymous communication systems to simultaneously address infrastructure independence, wire-format unobservability, and practical performance.

Nethr addresses this gap through three mechanisms. Symmetric trial decryption provides recipient anonymity without transmitting any identifier, the Poly1305 AEAD authentication tag serves as an implicit recipient identifier visible only to the session's legitimate participant. Identifierless BLAKE3-based deduplication eliminates message tracking across relay nodes by computing the deduplication key locally from the encrypted body, ensuring the key never traverses the network. BLAKE3-masked header obfuscation derives a per-packet header mask from the encrypted body, producing a wire format in which every byte, including the header, is computationally indistinguishable from uniform random noise. These mechanisms address the specific metadata and fingerprinting vulnerabilities identified in all surveyed prior systems.

Empirical evaluation confirms Nethr's core claims. The bounded epidemic routing algorithm sustains message delivery above 90% under 40% random node failure at fanout $k = 3$, and

above 99% at $k = 5$, meeting our stated delivery hypothesis across all tested scenarios. A notable emergent finding is that targeted hub removal degrades delivery less than random node failure of equivalent magnitude. This is an inversion of classical Barabási-Albert network vulnerability that we explain through the hub-as-bottleneck mechanism: hub nodes are routing bottlenecks under bounded epidemic routing, and their removal homogenizes the graph in a way that actually improves epidemic spreading efficiency. Statistical analysis of 1,000 live-captured Nethr packets confirms that the wire format is mathematically indistinguishable from ideal random noise under all four applied tests, with Chi-square p-values approximating 0.5 and Shannon entropy of $\sim$8 bits per byte across all packet positions. Furthermore, high-volume testing (Experiment F) empirically bounded the network throughput, identifying symmetric trial decryption compute as the primary scaling constraint.

The hybrid X25519 + sntrup761 key exchange synthesized within the Noise_XX handshake framework provides both classical and post-quantum forward secrecy. Session keys derived from this construction remain secure against an adversary who records traffic today and later acquires quantum computing capability, provided at least one of the two primitives remains secure.

We acknowledge the limitations of the current architecture. Nethr trades absolute, cryptographically-proven sender isolation for probabilistic plausible deniability. It does not provide formal differential privacy bounds, nor does it protect against a global active adversary performing intersection attacks over long observation windows. The bootstrap privacy problem remains an open engineering challenge.

Future work will pursue several critical directions: formal differential privacy analysis of bounded epidemic routing in the Universal Composability (UC) framework, Sybil admission control through lightweight proof-of-work, and the exploration of variable-sized UDP datagrams to further obscure statistical flow analysis. Crucially, the protocol daemon will migrate from the current C++20 Proof of Concept to a production-grade Rust implementation, resolving concurrency bottlenecks to maximize trial decryption throughput.

Nethr demonstrates that infrastructure-independent, DPI-resistant, post-quantum-secure communication is achievable with low latency on commodity hardware. The combination of properties it achieves provides a robust, decentralized foundation for resilient data transfer under adversarial conditions representative of real-world censorship environments.

## References

[1] Access Now, *Rising Repression Meets Global Resistance: Internet Shutdowns in 2025*, Access Now #KeepItOn coalition, 2026. Available: `https://www.accessnow.org/wp-content/uploads/2026/03/KeepItOn-Internet-Shutdowns-2025-Annual-Report.pdf`

[2] Freedom House, *Freedom on the Net 2025: An Uncertain Future for the Global Internet*, Freedom House, Washington D.C., 2025. Available: `https://freedomhouse.org/report/freedom-net/2025/uncertain-future-global-internet`

[3] Zona Media, *The 16-Kilobyte Curtain*, 2025. Available: `https://en.zona.media/article/2025/06/19/cloudflare`

[4] M. K. Goldschlag, M. G. Reed, P. F. Syverson, "Hiding Routing Information," in *Proc. 1st International Workshop on Information Hiding*, Cambridge, UK, Springer LNCS vol. 1174, 1996, pp. 137–150.

[5] R. Dingledine, N. Mathewson, P. Syverson, "Tor: The Second-Generation Onion Router," in *Proc. 13th USENIX Security Symposium*, San Diego, CA, 2004, pp. 303–320.

[6] P. Winter, T. Pulls, J. Fuß, "ScrambleSuit: Resistance to Active Probing and Passive Fingerprinting," in *Proc. 12th ACM Workshop on Privacy in the Electronic Society (WPES)*, Berlin, 2013, pp. 213–224.

[7] B. Zantout, R. Haraty, "I2P Data Communication System," in *Proc. 10th International Conference on Networks (ICN)*, St. Maarten, 2011, pp. 401–409.

[8] I2P Project, *I2P 2.11.0 Release*, Feb. 2026. Available: `https://geti2p.net/en/blog/post/2026/02/09/2.11.0-Release`

[9] J. Warren, *Bitmessage: A Peer-to-Peer Message Authentication and Delivery System*, 2012. Available: `https://bitmessage.org/bitmessage.pdf`

[10] A. Upadhyay, "EFPIX: A Zero-Trust Encrypted Flood Protocol for Infrastructure-Independent Communication," *arXiv preprint* arXiv:2509.08248, Sep. 2025. Available: `https://arxiv.org/abs/2509.08248`

[11] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.

[12] J. van den Hooff, D. Lazar, M. Zaharia, N. Zeldovich, "Vuvuzela: Scalable Private Messaging Resistant to Traffic Analysis," in *Proc. 25th ACM Symposium on Operating Systems Principles (SOSP)*, Monterey, CA, 2015, pp. 137–152.

[13] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, G. Danezis, "The Loopix Anonymity System," in *Proc. 26th USENIX Security Symposium*, Vancouver, BC, 2017, pp. 1199–1216.

[14] National Institute of Standards and Technology, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*, FIPS 203, U.S. Department of Commerce, Aug. 2024.

[15] D. J. Bernstein, C. Chuengsatiansup, T. Lange, C. van Vredendaal, "NTRU Prime: Reducing Attack Surface at Low Cost," in *Proc. Selected Areas in Cryptography (SAC 2017)*, Ottawa, ON, Springer LNCS vol. 10719, 2018, pp. 235–260.

[16] The OpenSSH Project, *OpenSSH 8.5 Release Notes*, Apr. 2021. Available: `https://www.openssh.com/releasenotes.html#8.5`

[17] H. Krawczyk, P. Eronen, *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*, RFC 5869, Internet Engineering Task Force, May 2010. Available: `https://www.rfc-editor.org/rfc/rfc5869`

[18] T. Perrin, *The Noise Protocol Framework*, Revision 34, Jul. 2018. Available: `https://noiseprotocol.org/noise.html`

[19] F. Denis, *libsodium Documentation*, version 1.0.21, 2026. Available: `https://doc.libsodium.org`

[20] The Open Quantum Safe Project, *liboqs 0.15.0*, Nov. 2025. Available: `https://github.com/open-quantum-safe/liboqs/releases/tag/0.15.0`

[21] A.-L. Barabási, R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[22] D. J. Watts, S. H. Strogatz, "Collective Dynamics of 'Small-World' Networks," *Nature*, vol. 393, pp. 440–442, Jun. 1998.

[23] R. Albert, H. Jeong, A.-L. Barabási, "Error and Attack Tolerance of Complex Networks," *Nature*, vol. 406, pp. 378–382, Jul. 2000.

*Email address*: `kripticni@proton.me`