1 RFC-0001: RFC Life Cycle, Process and Structure

- **RFC Number:** 0001
- **Title:** RFC Life Cycle, Process and Structure
- **Status:** Raw
- **Author(s):** Qianchen Yu (@QYuQianchen), Tino Breddin (@tolbrino)
- **Created:** 2025-02-20
- **Updated:** 2025-08-20
- **Version:** v0.2.0 (Raw)
- **Supersedes:** N/A
- **Related Links:** none

1.1 1. Abstract

This RFC defines the life cycle, contribution process, versioning system, governance model, and document structure for RFCs at HOPR. It outlines stages, naming conventions, validation rules, and formatting standards that MUST be followed to ensure consistency and clarity across all RFC submissions. The process ensures iterative development with feedback loops and transparent updates with pull requests (PR).

1.2 2. Motivation

HOPR project requires a clear and consistent process for managing technical proposals, documenting protocol architecture. A well-defined life cycle MUST be established to maintain coherence, ensure quality, and streamline future development.

1.3 3. Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [01].

Draft: An RFC is considered a draft from the moment it is proposed for review. A draft MUST include a clear summary, context, and initial technical details. Drafts MUST follow the v0.x.x versioning scheme, with each version being independently implementable. A draft version is assigned as soon as the first PR is created.

1.4 4. Specification

1.4.1 4.1. RFC Life Cycle Stages

4.1.1. Mermaid Diagram for RFC Life Cycle Stages4.1.1. Mermaid Diagram for RFC Life Cycle Stages

4.1.2. Stage Descriptions:4.1.2. Stage Descriptions:

- **Raw:** The RFC **MUST** begin as a raw draft reflecting initial ideas. The draft MAY contain incomplete details but MUST provide a clear objective.
- **Discussion:** Upon submission of the initial PR, the RFC number and `v0.1.0` version are assigned. Feedback SHALL be gathered via PRs, with iterative updates reflected in version increments `(v0.x.x)`.
- **Review:** The RFC **MUST** undergo at least one review cycle. The draft **SHOULD** incorporate significant feedback and each iteration **MUST** be independently implementable.
- **Draft:** The RFC moves into active development and refinement. Each update **SHALL** increment the version (`v0.x.x`) to indicate progress.
- **Implementation:** Merging to the main branch signifies readiness for practical use, triggering the finalization process.
- **Finalized:** The RFC is considered stable and complete, with version `v1.0.0` assigned. Only errata modifications are permitted afterward.
- **Errata:** Minor technical corrections post-finalization **MUST** be documented and result in a patch version increment (`v1.0.x`). Errata are technical corrections or factual updates made after an RFC has been finalized. They **MUST NOT** alter the intended functionality or introduce new features.
- **Superseded:** Significant updates requiring functionality changes **MUST** be documented in a new RFC, starting at `v2.0.0` or higher. The original RFC must include information that it has been superseded, accompanied with a link to the new RFC that supersedes it.
- **Rejected:** If an RFC does not progress past the discussion stage, reasons **MUST** be documented.

1.4.2 4.2. File Structure

[] RFC-0001-rfc-life-cycle-process/ 0001-rfc-life-cycle-process.md errata/ 0001-v1.0.1-erratum.md assets/ life-cycle-overview.png

---

1.4.3 4.3. Validation Rules

- Directory **MUST** be prefixed with uppercased "RFC", followed by its RFC number, and a succinct title all in lowercase joined by hyphens. E.g. `RFC-0001-rfc-life-cycle-process`
- Main file **MUST** be prefixed with its RFC number and a succinct title all in lowercase joined by hyphens. E.g. `0001-rfc-life-cycle-process.md`
- All assets **MUST** reside in the `assets/` folder.
- Errata **MUST** reside in the `errata/` folder.

1.4.4 4.4. RFC Document Structure

All RFCs **MUST** follow a consistent document structure to ensure readability and maintainability.

4.4.1. Metadata Preface Every RFC **MUST** begin with the following metadata structure:

[] # RFC-XXXX: [Title]

- **RFC Number:** XXXX - **Title:** [Title in Title Case] - **Status:** Raw — Discussion — Review — Draft — Implementation — Finalized — Errata — Rejected — Superseded - **Author(s):** [Name (GitHub Handle)] - **Created:** YYYY-MM-DD - **Updated:** YYYY-MM-DD - **Version:** vX.X.X (Status) - **Supersedes:** RFC-YYYY (if applicable) — N/A - **Related Links:** [RFC-XXXX](../RFC-XXXX-[slug]/XXXX-[slug].md) — none

4.4.2. Reference Styles RFCs **MUST** use two distinct reference styles:

4.4.2.1. RFC-to-RFC References

- RFC references to other HOPR RFCs **MUST** be listed in the metadata's **Related Links:** field
- Format: `[RFC-XXXX](../RFC-XXXX-[slug]/XXXX-[slug].md)`
- Multiple references **SHALL** be separated by commas
- If no RFC references exist, the field **MUST** contain "none"
- Example: `[RFC-0002](../RFC-0002-mixnet-keywords/0002-mixnet-keywords.md), [RFC-0004](../RFC-0004-hopr-packet-protocol/0004-hopr-packet-protocol.md)`

4.4.2.2. External References

- External references **MUST** be listed in a dedicated `## References` section at the end of the document
- References **MUST** use sequential numbering with zero-padding: [01], [02], etc.
- In-text citations **MUST** use the numbered format: "as described in [01]"
- Reference format **SHOULD** follow academic citation style:

```
[XX] Author(s). (Year). [Title](URL). _Publication_, Volume(Issue), pages.
```

- Example:

```
[01] Chaum, D. (1981). [Untraceable Electronic Mail, Return Addresses, and Digital P
```

4.4.3. Required Sections All RFCs **MUST** include the following sections:

1. **Metadata Preface** (as defined in 4.4.1)
2. **Abstract** - Brief summary of the RFC's purpose and scope
3. **References** - External citations (if any)

1.5 5. Design Considerations

- Modular RFCs **SHOULD** be preferred.
- PR system **MUST** be the primary mechanism for contribution, review, and errata handling.

1.6 6. Compatibility

- New RFCs **MUST** maintain backward compatibility unless explicitly stated.
- Errata **MUST NOT** introduce backward-incompatible changes.
- Breaking changes **MUST** be reflected in a major version increment (`v2.0.0`).

1.7 7. Security Considerations

- Security review phase **MUST** be included before finalization.
- Errata **MUST** undergo security review if impacting critical components.

1.8 8. Drawbacks

- Strict naming conventions **MAY** limit creative flexibility.

1.9 9. Alternatives

- Collaborative document editing tools, e.g. hackmd.

1.10 10. Unresolved Questions

- Handling emergency RFCs
- Enforcing cross-RFC dependencies
- Formal approval timeline for errata

1.11 11. Future Work

- Automated validation tools
- CI/CD integration for automated versioning and errata checks
- Web interface for publishing RFCs

1.12 12. References

[01] Bradner, S. (1997). https://datatracker.ietf.org/doc/html/rfc2119Key words for use in RFCs to Indicate Requirement Levels. *IETF RFC 2119*.

[02] https://www.rfc-editor.org/styleguide/RFC Editor Style Guide. RFC Editor.

[03] https://github.com/rust-lang/rfcsRust RFC Process. Rust Language Team.

[04] https://rfc.zeromq.orgZeroMQ RFC Process. ZeroMQ Community.

[05] https://github.com/vacp2p/rfc-indexVACP2P RFC Index. Vac Research.

2 RFC-0002: Common mixnet terms and keywords

- **RFC Number:** 0002
- **Title:** Common mixnet terms and keywords
- **Status:** Draft
- **Author(s):** Tino Breddin (@tolbrino)
- **Created:** 2025-08-01
- **Updated:** 2025-09-04
- **Version:** v0.1.0 (Draft)
- **Supersedes:** none
- **Related Links:** none

2.1 1. Abstract

This RFC provides a glossary of common terms and keywords related to mixnets and the HOPR protocol specifically. It aims to establish a shared vocabulary for developers, researchers, and users involved in the HOPR project.

2.2 2. Motivation

The HOPR project involves a diverse community of people with different backgrounds and levels of technical expertise. A shared vocabulary is essential for clear communication and a common understanding of the concepts and technologies used in the project. This RFC aims to provide a single source of truth for the terminology used in the HOPR ecosystem.

2.3 3. Terminology

- **Mixnet:** Also known as a **Mix network** is a routing protocol that creates hard-to-trace communications by using a chain of proxy servers known as mixes which take in messages from multiple senders, shuffle them, and send them back out in random order to the next destination.
- **Node:** A process which implements the HOPR protocol and participates in the mixnet. Nodes can be run by anyone. A node can be a sender, destination or a relay node which helps to relay messages through the network. Also referred to as "peer" [01, 02].
- **Sender:** The node that initiates communication by sending out a packet through the mixnet. This is typically an application which wants to send a message anonymously [01, 02].

- **Destination:** The node that receives a message sent through the mixnet. Also referred to as "receiver" in some contexts [01, 02].
- **Peer**: A node that is connected to another node in the p2p network. Each peer has a unique identifier and can communicate with other peers. The terms "peer" and "node" are often used interchangeably.
- **Cover Traffic:** Artificial data packets introduced into the network to obscure traffic patterns with adaptive noise. These data packets can be generated on any node and are used to make it harder to distinguish between real user traffic and dummy traffic [01, 03].
- **Path:** The route a message takes through the mixnet, defined as a sequence of hops between sender and destination. A path can be direct from sender to destination, or it can go through multiple relay nodes before reaching the destination. Also referred to as "message path" [01, 02].
- **Forward Path:** A path that is used to deliver a packet only in the direction from the sender to the destination.
- **Return Path:** A path that is used to deliver a packet in the opposite direction than the forward path. The return path MAY be disjoint with the forward path.
- **Relay Node:** A node that forwards messages from one node to another in the mixnet. Relay nodes help to obscure the sender's identity by routing messages through multiple nodes [01, 02].
- **Hop:** A relay node in the message path that is neither the sender nor the destination. E.g. a 0-hop message is sent directly from the sender to the destination, while a 1-hop message goes through one relay node before reaching the destination. The terms "hop" and "relay" are often used interchangeably [01, 02]. More hops in the path generally increase the anonymity of the message, but also increase latency and cost.
- **Mix Nodes:** These are the proxy servers that make up the mixnet. They receive messages from multiple senders, shuffle them, and then send them back out in a random order [01].
- **Layered Encryption:** A technique where a message is wrapped in successive layers of encryption. Each intermediary node (or hop) can only decrypt its corresponding layer, revealing the next destination in the path [01, 04].
- **Metadata:** Data that provides information about other data. In the context of mixnets, this includes things like the sender's and destination's IP addresses, the size of the message, and the time it was sent or received. Mixnets work to shuffle this metadata to protect user privacy [01, 06].

- **Onion Routing:** A technique for anonymous communication over a network. It involves encrypting messages in layers, analogous to the layers of an onion, which are then routed through a series of network nodes [04].
- **Public Key Cryptography:** A cryptographic system that uses pairs of keys: public keys, which may be disseminated widely, and private keys, which are known only to the owner. This is used to encrypt messages sent through the mixnet [01].
- **Sphinx:** A packet format that ensures unlinkability and layered encryption. It uses a fixed-size packet structure to resist traffic analysis [02].
- **Symmetric Encryption:** A type of encryption where the same key is used to both encrypt and decrypt data [05].
- **Traffic Analysis:** The process of intercepting and examining messages in order to deduce information from patterns in communication. Mixnets are designed to make traffic analysis very difficult [01].
- **Forward Message:** A packet that is sent along the forward path. Also referred to as "forward packet".
- **Reply Message:** A packet that is sent along the return path. Also referred to as "reply packet".

2.4 4. References

[01] Chaum, D. (1981). https://www.freehaven.net/anonbib/cache/chaum-mix.pdfUntraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM, 24*(2), 84-90.

[02] Danezis, G., & Goldberg, I. (2009). https://cypherpunks.ca/ iang/pubs/Sphinx$_{O}akland09.pdfSphi$ 277.

[03] K. Sampigethaya and R. Poovendran, A Survey on Mix Networks and Their Secure Applications. Proceedings of the IEEE, vol. 94, no. 12, pp. 2142-2181, Dec. 2006.

[04] Reed, M. G., Syverson, P. F., & Goldschlag, D. M. (1998). https://www.onion-router.net/Publications/JSAC-1998.pdfAnonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications, 16*(4), 482-494.

[05] Shannon, C. E. (1949). Communication Theory of Secrecy Systems. *Bell System Technical Journal, 28*(4), 656-715. DOI: 10.1002/j.1538-7305.1949.tb00928.x

[06] Cheu, A., Smith, A., Ullman, J., Zeber, D., & Zhilyaev, M. (2019, April). Distributed differential privacy via shuffling. In Annual international conference on the theory and applications of cryptographic techniques (pp. 375-403). Cham: Springer International Publishing.