Mix Network Specification

Yawning Angel
George Danezis
Claudia Diaz
Ania Piotrowska
David Stainton

Table of Contents

Abstract

This document describes the high level architecture and detailed protocols and behavior required of mix nodes participating in the Katzenpost Mix Network.

1. Introduction

This specification provides the design of a mix network meant provide an anonymous messaging protocol between clients and public mixnet services.

Various system components such as client software, end to end messaging protocols, Sphinx cryptographic packet format and wire protocol are described in their own specification documents.

1.1 Terminology

- A `KiB` is defined as 1024 8 bit octets.
- `Mixnet` - A mixnet also known as a mix network is a network of mixes that can be used to build various privacy preserving protocols.
- `Mix` - A cryptographic router that is used to compose a mixnet. Mixes use a cryptographic operation on messages being routed which provides bitwise unlinkability with respect to input versus output messages. Katzenpost is a decryption mixnet that uses the Sphinx cryptographic packet format.
- `Node` - A Mix. Client's are NOT considered nodes in the mix network. However note that network protocols are often layered; in our design documents we describe "mixnet hidden services" which can be operated by mixnet clients. Therefore if you are using node in some adherence to methematical termonology one could conceivably designate a client as a node. That having been said, it would not be appropriate to the discussion of our core mixnet protocol to refer to the clients as nodes.
- `Entry mix`, `Entry node` - An entry mix is a mix that has some additional features:

1. An entry mix is always the first hop in routes where the message originates from a client.
2. An entry mix authenticates client’s direct connections via the mixnet’s wire protocol.
3. An entry mix queues reply messages and allows clients to retrieve them later.

- `Service mix` - A service mix is a mix that has some additional features:

1. A service mix is always the last hop in routes where the message originates from a client.
2. A service mix runs mixnet services which use a Sphinx SURB based protocol.

- `User` - An agent using the Katzenpost system.
- `Client` - Software run by the User on its local device to participate in the Mixnet. Again let us reiterate that a client is not considered a "node in the network" at the level of analysis where we are discussing the core mixnet protocol in this here document.
- `Katzenpost` - A project to design many improved decryption mixnet protocols.

  Classes of traffic - We distinguish the following classes of traffic:

- SURB Replies (also sometimes referred to as ACKs)
- Forward messages
- `Packet` - Also known as a Sphinx packet. A nested encrypted packet that, is routed through the mixnet and cryptographically transformed at each hop. The length of the packet is fixed for every class of traffic. Packet payloads encapsulate messages.
- `Payload` - The payload, also known as packet payload, is a portion of a Packet containing a message, or part of a message, to be delivered anonymously.
- `Message` - A variable-length sequence of octets sent anonymously through the network. Short messages are sent in a single packet; long messages are fragmented across multiple packets.
- `MSL` - Maximum Segment Lifetime, 120 seconds.

1.2 Conventions Used in This Document

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 RFC2119

2. System Overview

The presented system design is based on LOOPIX Below, we present the system overview.

The entry mixes are responsible for authenticating clients, accepting packets from the client, and forwarding them to the mix network, which then relays packets to the destination service mix. Our network design uses a strict topology where forward message traverse the network from entry mix to service mix. Service mixes can optionally reply if the forward message contained a Single Use Reply Block (see SPHINXSPEC.

The PKI system that handles the distribution of various network wide parameters, and information required for each participant to participate in the network such as IP address/port combinations that each node can be reached at, and cryptographic public keys. The specification for the PKI is beyond the scope of this document and is instead covered in KATZMIXPKI.

The mix network provides neither reliable nor in-order delivery semantics. The described mix network is neither a user facing messaging system nor is it an application. It is intended to be a low level protocol which can be composed to form more elaborate mixnet protocols with stronger more useful privacy notions.

2.1 Threat Model

Here we cannot present the threat model to the higher level mixnet protocols. However this low level core mixnet protocol does have it's own threat model which we attempt to illucidate here.

We assume that the clients only talk to mixnet services. These services make use of a client provided delivery token known as a SURB (Single Use Reply Block) to send their replies to the client without knowing the client's entry mix. This system guarantees third-party anonymity, meaning that no parties other than client and the service are able to learn that the client and service are communicating. Note that this is in contrast with other designs, such as Mixminion, which provide sender anonymity towards recipients as well as anonymous replies.

Mixnet clients will randomly select an entry node to use and may reconnect if disconnected for under a duration threshold. The entry mix can determine the approximate message volume originating from and destined to a given client. We consider the entry mix follows the protocol and might be an honest-but-curious adversary.

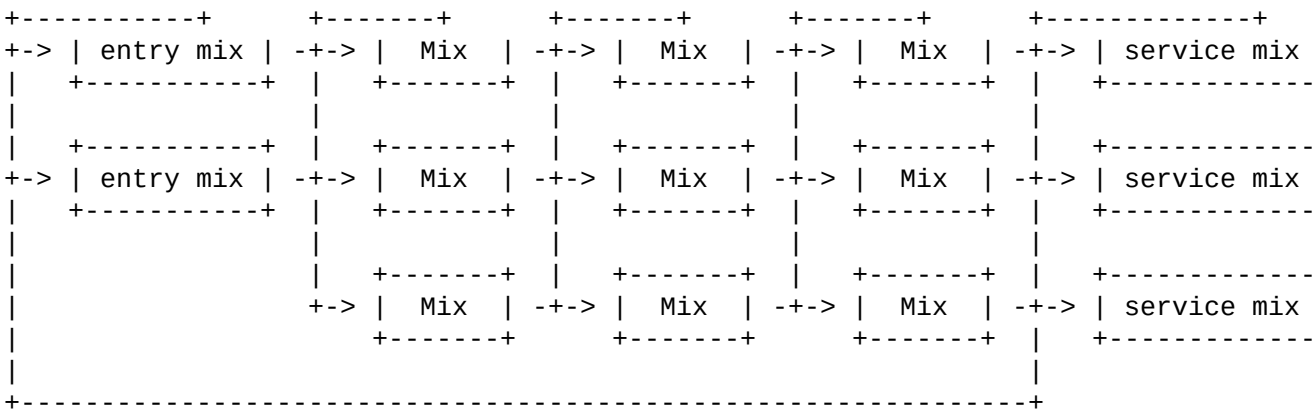
External local network observers can not determine the number of Packets traversing their region of the network because of the use of decoy traffic sent by the clients. Global observers will not be able to de-anonymize packet paths if there are enough packets traversing the mix network. Longer term statistical disclosure attacks are likely possible in order to link senders and receivers.

A malicious mix only has the ability to remember which input packets correspond to the output packets. To discover the entire path all of the mixes in the path would have to be malicious. Moreover, the malicious mixes can drop, inject, modify or delay the packets for more or less time than specified.

2.2 Network Topology

The Katzenpost Mix Network uses a layered topology consisting of a fixed number of layers, each containing a set of mixes. At any given time each Mix MUST only be assigned to one specific layer. Each Mix in a given layer N is connected to every other Mix in the previous and next layer, and or every participating Provider in the case of the mixes in layer 0 or layer N (first and last layer). :

```
Layer 0          Layer 1        Layer 2        Layer 3          Layer 4
```



Note: Multiple distinct connections are collapsed in the figure for sake of brevity/clarity.

The network topology MUST also maximize the number of security domains traversed by the packets. This can be achieved by not allowing mixes from the same security domain to be in different layers.

Requirements for the topology:

- Should allow for non-uniform throughput of each mix (Get bandwidth weights from the PKI).
- Should maximize distribution among security domains, in this case the mix descriptor specified family field would indicate the security domain or entity operating the mix.
- Other legal jurisdictional region awareness for increasing the cost of compulsion attacks.

3. Packet Format Overview

For the packet format of the transported messages we use the Sphinx cryptographic packet format. The detailed description of the packet format, construction, processing and security / anonymity considerations see SPHINXSPEC, “The Sphinx Mix Network Cryptographic Packet Format Specification”.

As the Sphinx packet format is generic, the Katzenpost Mix Network must provide a concrete instantiation of the format, as well as additional Sphinx per-hop routing information commands.

3.1 Sphinx Cryptographic Primitives

For the current version of the Katzenpost Mix Network, let the following cryptographic primitives be used as described in the Sphinx specification.

- `H(M)` - As the output of this primitive is only used locally to a Mix, any suitable primitive may be used.
- `MAC(K, M)` - HMAC-SHA256 RFC6234, M_KEY_LENGTH of 32 bytes (256 bits), and MAC_LENGTH of 32 bytes (256 bits).
- `KDF(SALT, IKM)` - HKDF-SHA256, HKDF-Expand only, with SALT used as the info parameter.
- `S(K, IV)` - CTR-AES256 [SP80038A], S_KEY_LENGTH of 32 bytes (256 bits), and S_IV_LENGTH of 12 bytes (96 bits), using a 32 bit counter.
- `SPRP_Encrypt(K, M)/SPRP_Decrypt(K, M)` - AEZv5 AEZV5, SPRP_KEY_LENGTH of 48 bytes (384 bits). As there is a disconnect between AEZv5 as specified and the Sphinx usage, let the following be the AEZv5 parameters:
  - nonce - 16 bytes, reusing the per-hop Sphinx header IV.
  - additional_data - Unused.
  - tau - 0 bytes.
- `EXP(X, Y)` - X25519 RFC7748 scalar multiply, GROUP_ELEMENT_LENGTH of 32 bytes (256 bits), G is the X25519 base point.

3.2 Sphinx Packet Parameters

The following parameters are used as for the Katzenpost Mix Network instantiation of the Sphinx Packet Format:

- `AD_SIZE` - 2 bytes.
- `SECURITY_PARAMETER` - 32 bytes. (except for our SPRP which we plan to upgrade)
- `PER_HOP_RI_SIZE` - (XXX/ya: Addition is hard, let's go shopping.)
- `NODE_ID_SIZE` - 32 bytes, the size of the Ed25519 public key, used as Node identifiers.
- `RECIPIENT_ID_SIZE` - 64 bytes, the maximum size of local-part component in an e-mail address.
- `SURB_ID_SIZE` - Single Use Reply Block ID size, 16 bytes.
- `MAX_HOPS` - 5, the ingress provider, a set of three mixes, and the egress provider.
- `PAYLOAD_SIZE` - (XXX/ya: Subtraction is hard, let's go shopping.)
- `KDF_INFO` - The byte string `Katzenpost-kdf-v0-hkdf-sha256`.

The Sphinx Packet Header `additional_data` field is specified as follows:

```
struct {
    uint8_t version;  /* 0x00 */
    uint8_t reserved; /* 0x00 */
} KatzenpostAdditionalData;
```

Double check to ensure that this causes the rest of the packet header to be 4 byte aligned, when wrapped in the wire protocol command and framing. This might need to have 3 bytes reserved instead.

All nodes MUST reject Sphinx Packets that have `additional_data` that is not as specified in the header.

Design decision.

- We can eliminate a trial decryption step per packet around the epoch transitions by having a command that rewrites the AD on a per-hop basis and including an epoch identifier.

I am uncertain as to if the additional complexity is worth it for a situation that can happen for a few minutes out of every epoch.

3.3 Sphinx Per-hop Routing Information Extensions

The following extensions are added to the Sphinx Per-Hop Routing Information commands.

Let the following additional routing commands be defined in the extension RoutingCommandType range (`0x80` - `0xff`):

```
enum {
    mix_delay(0x80),
} KatzenpostCommandType;
```

The mix_delay command structure is as follows:

```
struct {
    uint32_t delay_ms;
} NodeDelayCommand;
```

4. Mix Node Operation

All Mixes behave in the following manner:

- Accept incoming connections from peers, and open persistent connections to peers as needed `Section 4.1 <4.1>`.
- Periodically interact with the PKI to publish Identity and Sphinx packet public keys, and to obtain information about the peers it should be communicating with, along with periodically rotating the Sphinx packet keys for forward secrecy `Section 4.2 <4.2>`.
- Process inbound Sphinx Packets, delay them for the specified time and forward them to the appropriate Mix and or Provider `Section 4.3 <4.3>`.

All Nodes are identified by their link protocol signing key, for the purpose of the Sphinx packet source routing hop identifier.

All Nodes participating in the Mix Network MUST share a common view of time, via NTP or similar time synchronization mechanism.

4.1 Link Layer Connection Management

All communication to and from participants in the Katzenpost Mix Network is done via the Katzenpost Mix Network Wire Protocol KATZMIXWIRE.

Nodes are responsible for establishing the connection to the next hop, for example, a mix in layer 0 will accept inbound connections from all Providers listed in the PKI, and will proactively establish connections to each mix in layer 1.

Nodes MAY accept inbound connections from unknown Nodes, but MUST not relay any traffic until they became known via listing in the PKI document, and MUST terminate the connection immediately if authentication fails for any other reason.

Nodes MUST impose an exponential backoff when reconnecting if a link layer connection gets terminated, and the minimum retry interval MUST be no shorter than 5 seconds.

Nodes MAY rate limit inbound connections as required to keep load and or resource use at a manageable level, but MUST be prepared to handle at least one persistent long lived connection per potentially eligible peer at all times.

4.2 Sphinx Mix and Provider Key Rotation

Each Node MUST rotate the key pair used for Sphinx packet processing periodically for forward secrecy reasons and to keep the list of seen packet tags short. The Katzenpost Mix Network uses a fixed interval (`epoch`), so that key rotations happen simultaneously throughout the network, at predictable times.

Let each epoch be exactly `10800 seconds (3 hours)` in duration, and the 0th Epoch begin at `2017-06-01 00:00 UTC`. For more details see our “Katzenpost Mix Network Public Key Infrastructure Specification” document. KATZMIXPKI

4.3 Sphinx Packet Processing

The detailed processing of the Sphinx packet is described in the Sphinx specification: “The Sphinx Mix Network Cryptographic Packet Format Specification”. Below, we present an overview of the steps which the node is performing upon receiving the packet:

1. Records the time of reception.
2. Perform a `Sphinx_Unwrap` operation to authenticate and decrypt a packet, discarding it immediately if the operation fails.
3. Apply replay detection to the packet, discarding replayed packets immediately.
4. Act on the routing commands.

All packets processed by Mixes MUST contain the following commands.

- `NextNodeHopCommand`, specifying the next Mix or Provider that the packet will be forwarded to.
- `NodeDelayCommand`, specifying the delay in milliseconds to be applied to the packet, prior to forwarding it to the Node specified by the NextNodeHopCommand, as measured from the time of reception.

Mixes MUST discard packets that have any commands other than a `NextNodeHopCommand` or a `NodeDelayCommand`. Note that this does not apply to Providers or Clients, which have additional commands related to recipient and `SURB (Single Use Reply Block)` processing.

Nodes MUST continue to accept the previous epoch's key for up to 1MSL past the epoch transition, to tolerate latency and clock skew, and MUST start accepting the next epoch's key 1MSL prior to the epoch transition where it becomes the current active key.

Upon the final expiration of a key (1MSL past the epoch transition), Nodes MUST securely destroy the private component of the expired Sphinx packet processing key along with the backing store used to maintain replay information associated with the expired key.

Nodes MAY discard packets at any time, for example to keep congestion and or load at a manageable level, however assuming the `Sphinx_Unwrap` operation was successful, the packet MUST be fed into the replay detection mechanism.

Nodes MUST ensure that the time a packet is forwarded to the next Node is around the time of reception plus the delay specified in `NodeDelayCommand`. Since exact millisecond processing is unpractical, implementations MAY tolerate a small window around that time for packets to be forwarded. That tolerance window SHOULD be kept minimal.

Nodes MUST discard packets that have been delayed for significantly more time than specified by the `NodeDelayCommand`.

5. Anonymity Considerations

5.1 Topology

Layered topology is used because it offers the best level of anonymity and ease of analysis, while being flexible enough to scale up traffic. Whereas most mixnet papers discuss their security properties in the context of a cascade topology, which does not scale well, or a free-route network, which quickly becomes intractable to analyze when the network grows, while providing slightly worse anonymity than a layered topology. MIXTOPO10

Important considerations when assigning mixes to layers, in order of decreasing importance, are:

1. Security: do not allow mixes from one security domain to be in different layers to maximise the number of security domains traversed by a packet
2. Performance: arrange mixes in layers to maximise the capacity of the layer with the lowest capacity (the bottleneck layer)
3. Security: arrange mixes in layers to maximise the number of jurisdictions traversed by a packet (this is harder to do really well than it seems, requires understanding of legal agreements such as MLATs).

5.2 Mixing strategy

As a mixing technique the Poisson mix strategy LOOPIX and KESDOGAN98 is used, which REQUIRES that a packet at each hop in the route is delayed by some amount of time, randomly selected by the sender from an exponential distribution. This strategy allows to prevent the timing correlation of the incoming and outgoing traffic from each node. Additionally, the parameters of the distribution used for generating the delay can be tuned up and down depending on the amount of traffic in the network and the application for which the system is deployed.

6. Security Considerations

The source of all authority in the mixnet system comes from the Directory Authority system which is also known as the mixnet PKI. This system gives the mixes and clients a consistent view of the network while allowing human intervention when needed. All public mix key material and network connection information is distributed by this Directory Authority system.

Appendix A. References

Appendix A.1 Normative References

Appendix A.2 Informative References

Appendix B. Citing This Document

Appendix B.1 Bibtex Entry

Note that the following bibtex entry is in the IEEEtran bibtex style as described in a document called “How to Use the IEEEtran BIBTEX Style”.

```
@online{KatzMixnet,
title = {Katzenpost Mix Network Specification},
author = {Yawning Angel and George Danezis and Claudia Diaz and Ania Piotrowska an
url = {https://github.com/katzenpost/katzenpost/blob/main/docs/specs/mixnet.rst},
year = {2017}
}
```

AEZV5

```
Hoang, V., Krovetz, T., Rogaway, P.,
"AEZ v5: Authenticated Encryption by Enciphering",
March 2017,
http://web.cs.ucdavis.edu/~rogaway/aez/aez.pdf
```

KATZMIXE2E

```
Angel, Y., Danezis, G., Diaz, C., Piotrowska, A., Stainton, D.,
```

"Katzenpost Mix Network End-to-end Protocol Specification",
July 2017,
https://github.com/katzenpost/katzenpost/blob/main/docs/specs/old/end_to_end.md

KATZMIXPKI

Angel, Y., Piotrowska, A., Stainton, D.,
"Katzenpost Mix Network Public Key Infrastructure Specification",
December 2017,
https://github.com/katzenpost/katzenpost/blob/master/docs/specs/pki.md

KATZMIXWIRE

Angel, Y.,
"Katzenpost Mix Network Wire Protocol Specification",
June 2017.
https://github.com/katzenpost/katzenpost/blob/master/docs/specs/wire-protocol.md

KESDOGAN98

Kesdogan, D., Egner, J., and Büschkes, R.,
"Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System."
Information Hiding, 1998,
https://www.freehaven.net/anonbib/cache/stop-and-go.pdf

LOOPIX

Piotrowska, A., Hayes, J., Elahi, T., Meiser, S., Danezis, G.,
"The Loopix Anonymity System",
USENIX, August, 2017
https://arxiv.org/pdf/1703.00536.pdf

MIXTOPO10

Diaz, C., Murdoch, S., Troncoso, C.,
"Impact of Network Topology on Anonymity and Overhead in Low-Latency Anonymity Net
PETS, July 2010,
https://www.esat.kuleuven.be/cosic/publications/article-1230.pdf

RFC2119

Bradner, S.,
"Key words for use in RFCs to Indicate Requirement Levels",
BCP 14, RFC 2119, DOI 10.17487/RFC2119,
March 1997,
http://www.rfc-editor.org/info/rfc2119

RFC5246

Dierks, T. and E. Rescorla,
"The Transport Layer Security (TLS) Protocol Version 1.2",
RFC 5246, DOI 10.17487/RFC5246,
August 2008,
https://www.rfc-editor.org/info/rfc5246

RFC6234

Eastlake 3rd, D. and T. Hansen,
"US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)\"
RFC 6234, DOI 10.17487/RFC6234,
May 2011,
https://www.rfc-editor.org/info/rfc6234

RFC7748

Langley, A., Hamburg, M., and S. Turner,
"Elliptic Curves for Security",
RFC 7748,
January 2016.

SP80038A

Dworkin, M.,
"Recommendation for Block Cipher Modes of Operation",
SP800-38A, 10.6028/NIST.SP.800,
December 2001,
https://doi.org/10.6028/NIST.SP.800-38A

SPHINXSPEC

Angel, Y., Danezis, G., Diaz, C., Piotrowska, A., Stainton, D.,
"Sphinx Mix Network Cryptographic Packet Format Specification"
July 2017,
https://github.com/katzenpost/katzenpost/blob/master/docs/specs/sphinx.md