Katzenpost Administrator's Guide

Katzenpost Administrator's Guide

Table of Contents

List of Figures

List of Tables

Chapter 1. Components of the EchoMix mixnet

Directory authorities

Directory authorities, or *dirauths*, are nodes in EchoMix's decentralized public key infrastructure (PKI), which forms the security root of the entire EchoMix system. Clients and mixnet nodes rely on the PKI to maintain and sign an up-to-date *consensus* document, providing a view of the entire network including connection information and public cryptographic key materials and signatures.

Every 20 minutes (the current value for an *epoch*), each mix node signs a descriptor and uploads it to the dirauth nodes. The dirauths then vote on a new consensus document. If consensus is reached, each dirauth signs the document. Clients and nodes download the document as needed and verify the signatures. Consensus fails when 1/2 + 1 nodes fail, which yields greater fault tolerance than, for example, Byzantine Fault Tolerance, which fails when 1/3 + 1 of the nodes fail.

The PKI signature scheme is fully configurable by the dirauth nodes. Our recommendation is to use a hybrid signature scheme consisting of the classical Ed25519 and the post-quantum, stateless, hash-based signature scheme known as Sphincs+ (with the parameters: "sphincs-shake-256f"), which is designated in EchoMix configurations as "Ed25519 Sphincs+". Examples are provided below.

Configuring directory authorities

The following configuration is drawn from the reference implementation in `katzenpost/docker/voting_mixnet/auth1/authority.toml`. In a real-world mixnet, the component hosts would not be sharing a single IP address. For more information about the test mixnet, see Using the EchoMix test network.

Server section

```
[Server]
    Identifier = "auth1"
    WireKEMScheme = "xwing"
    PKISignatureScheme = "Ed25519 Sphincs+"
    Addresses = ["tcp://127.0.0.1:30001"]
    DataDir = "/voting_mixnet/auth1"
```

- **Identifier**

  Identifier is the human-readable identifier for a node, and should be unique per mixnet. It can be an FQDN but does not have to be.

  Type: string

  Required:

- **WireKEMScheme**

  WireKEMScheme specifies the key encapsulation mechanism (KEM) scheme for the PQ Noise [https://eprint.iacr.org/2022/539]-based wire protocol (link layer) that nodes use to communicate with each oth-

er. PQ Noise is a post-quantum variation of the Noise protocol framework [https://noiseprotocol.org/], which algebraically transforms ECDH handshake patterns into KEM encapsulate/decapsulate operations.

This configuration option supports the optional use of post-quantum cryptography to strengthen security. The following KEM schemes are supported:

- **Classical:** "x25519", "x448"
- **Post-quantum:** "mlkem768", "sntrup4591761", "frodo640shake", "mceliece348864", "mceliece348864f", "mceliece460896", "mceliece460896f", "mceliece6688128", "mceliece6688128f", "mceliece6960119", "mceliece6960119f", "mceliece8192128", "mceliece8192128f", "xwing", "Kyber768-X25519", "MLKEM768-X25519", "MLKEM768-X448", "CTIDH511", "CTIDH512", "CTIDH1024", "CTIDH2048", "CTIDH512-X25519", "CTIDH512-X25519"

Type: string

Required: Yes

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme which will be used by all components of the mix network when interacting with the PKI system. Mix nodes will sign their descriptors using this signature scheme. Likewise PKI documents will be signed by the dir auth nodes using this signature scheme.

  The following signature schemes are supported: "ed25519", "ed448", "Ed25519 Sphincs+", "Ed448-Sphincs+", "Ed25519-Dilithium2", "Ed448-Dilithium3"

  Type: string

  Required:

- **Addresses**

  Addresses specifies a list of one or more Address URLs in a format that contains the transport protocol, IP address and port number that the server will bind to for incoming connections. Currently EchoMix supports URLs with that start with either "tcp://" or "quic://" such as: ["tcp://192.168.1.1:30001"] and ["quic://192.168.1.1:40001"]

  Type: []string

  Required:

- **DataDir**

  DataDir is the absolute path to the directory authority server's state directory where the persistence.db will be written to disk AND where the server's cryptographic key materials will be written to disk when started with the "-g" commmandline option.

  Type: string

  Required:

Authorities section

An Authorities section is configured for each peer authority. We recommend using TOML's muli-line quotes for key materials.

```
[[Authorities]]
    Identifier = "auth1"
    IdentityPublicKey = """
-----BEGIN ED25519 PUBLIC KEY-----
dYpXpbozjFfqhR45ZC2q97SOOsXMANdHaEdXrP42CJk=
-----END ED25519 PUBLIC KEY-----
"""
    PKISignatureScheme = "Ed25519"
    LinkPublicKey = """
-----BEGIN XWING PUBLIC KEY-----
ooQBPYNdmfwnxXmvnljPA2mG5gWgurfHhbY87DMRY2tbMeZpinJ5BlSiIecprnmm
QqxcS9o36IS62SVMlOUkw+XEZGVvc9wJqHpgEgVJRAs1PCR8cUAdM6QIYLWt/lkf
SPKDCtZ3GiSIOzMuaglo2tarIPEv1AY7r9B0xXOgSKMkGyBkCfw1VBZf46MM26NL
opKx+rAKBzjKBJgGg9AHt9rNQsBbCVFBO+KKbpHHXFCBztK1Hhei3LeeYxcqL4bC
Rzi2kfBvQLPNzIuHOktKX7m39CxMByMNm4qL6dSVbtZaMOenpUlsIHOR5vyF2bgX
syRS3WnPRaIOarkMXUwJpzasNzuPUSAmJyaAQVwgrTSJyXp91usF6wBHKApqlDVY
dldqH7O90EI1tYNiOYpBCgwGBIK+xOkjoVNY91B/PVbA5ngcXgt8xQp02WXF1lVN
+ixu2XF8ndJXtLmlTIMUb0YwHsEo9UqdN7yLMPqcVScqR+ZKyCpaIPS+dpF20AAu
fvVVlxKVLqonUkJLCMgR/4morUOXVHwRgkKr0mIuw9RTWwSlSGxasFiVFZM5p6kS
ElIgPlCX94xV52BjcerPQ1R2TJdlVdWgLUAO7UNhyETCGrdxjxZSYmxNjfsOLHk7
qFNNDhUqHBjJmYOSDis0qHNXW8oZ9SF6pFN9NlctH8Uq10c/fNltykEbTkgLdMJP
phWWVQWD1YIU+/ktO5ZNVIVK/CHOqdc41+mRyxh/OmbEWMaaicuijLNR3KV9QoHG
9JWw9RtrPeE+jbgUX0g0asFf0nibfWqdV5QfAfwV/shYuIJR4nZMb6B4R9t8wYd0
gCSZg1M7gJW1J5B17cPM6+OCz2JaYhgAHfvJhSYK3zGzWWvGWvKTJrSTjiIhX8wM
H/KiX5U8+nkWPDiu/nqyRVej8/ELYvg7T2R7tKJzB0Gr5GUFd3B1TzvAcIRBl+yd
9ma5tRE44cu1yONuc1N8ITp8xqezSjwdavoMeuphryd6xbacZCVCcnM2R5mGOAhI
xzQWA5teHxoBTnliRQq1HriKX8N8Ogq3GPYu0Lue+MwEkkRkxSJdNpSLE/yzOQkf
DrlBbbURIfe1T6aeA1SuXeacTzqd6lfPWvjCgXMYriwhAazKwgS+WwcXd5RKbWEN
PIISgRavtoahEdOVSJJbVMUYptA3+CcY9eEi/rWOc1RgTWGNtWW0FYnFyiq6UdPM
NiO/z7S9uJJI/2wWMlJi+OoTx9cf3mjIMOPLkzEOhpVRhCZvcSvFIoF6ItcETnda
HmmOFwFg2reqlPioBORUufk7zdhs0MuuMYlaG+E0OqxTOKiZg0qbgMZqtOlsXkp8
QdY8R/W4ePiVAI2iZnMzEqMl3LLEDlRdMOkBYzOkR2GSvpYv6tetgoNAu8vFkpc6
Rma3y7KnihWaYhwEFbEqV2kD2GO9cMV+AcFvuCE9dVGVSatdxsSNX6ZXaFGlmHCp
gHtNyQJnXski52O03JpZRIhR40pFOhAAcMMAZDpMTVoxlcdR6WA4SlBiSceeJBgY
Yp9PlGhCimx9am99TrdLoLCdTHB6oowt8tss3POpIOxaSlguyeym/sBhkUrnXOgN
ldMtDsvvc9KUfE4I0+c+XQ==
-----END XWING PUBLIC KEY-----
    """
    WireKEMScheme = "xwing"
    Addresses = ["tcp://127.0.0.1:30001"]
```

- **Identifier**

  Identifier is the human readable identifier for the node which should be unique per mixnet. It can be an FQDN but certainly doesn't have to be.

  Type: string

  Required:

- **IdentityPublicKey**

  String in PEM format containing the public identity key. The node's public identity key is the node's permanent identifier and is used to verify cryptographic signatures produce by node's private identity key.

  Type: string

  Required:

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme used by all of the directory authority nodes. This PKISignatureScheme must match the scheme specified in the Server section of the configuration.

  Type: string

  Required:

- **LinkPublicKey**

  LinkPublicKeyPem is string containing the PEM format of the peer's public link layer key. The EchoMix "link layer" protocol is the PQ Noise based protocol which is sometimes also referred to as the "wire protocol". The type of this link public key must match the specified WireKEMScheme. Our wire protocol is used to secure the transport between nodes only and in this case it's used between directory authority nodes when they communicate with each other for the voting rounds and it's also the protocol that clients use to retrieve published PKI documents from the dir auth nodes.

  Type: string

  Required:

- **WireKEMScheme**

  WireKEMScheme specifies the key encapsulation mechanism (KEM) scheme for the PQ Noise [https://eprint.iacr.org/2022/539]-based wire protocol (link layer) that nodes use to communicate with each other. PQ Noise is a post-quantum variation of the Noise protocol framework [https://noiseprotocol.org/], which algebraically transforms ECDH handshake patterns into KEM encapsulate/decapsulate operations.

  This configuration option supports the optional use of post-quantum cryptography to strengthen security. The following KEM schemes are supported:

  - **Classical:** "x25519", "x448"

  - **Post-quantum:** "mlkem768", "sntrup4591761", "frodo640shake", "mceliece348864", "mceliece348864f", "mceliece460896", "mceliece460896f", "mceliece6688128", "mceliece6688128f", "mceliece6960119", "mceliece6960119f", "mceliece8192128", "mceliece8192128f", "xwing", "Kyber768-X25519", "MLKEM768-X25519", "MLKEM768-X448", "CTIDH511", "CTIDH512", "CTIDH1024", "CTIDH2048", "CTIDH512-X25519", "CTIDH512-X25519"

  Type: string

  Required: Yes

- **Addresses**

  Addresses specifies a list of one or more Address URLs in a format that contains the transport protocol, IP address and port number that the server will bind to for incoming connections. Currently EchoMix supports URLs with that start with either "tcp://" or "quic://" such as: ["tcp://192.168.1.1:30001"] and ["quic://192.168.1.1:40001"]

  Type: []string

  Required:

Logging section

The logging configuration section controls logging.

```
[Logging]
    Disable = false
    File = "katzenpost.log"
    Level = "INFO"
```

- **Disable**

  Disables logging if set to **true**.

  Type: bool

  Required:

- **File**

  Specifies the log file. If omitted, stdout is used.

  An absolute or relative file path can be specified. A relative path is relative to the DataDir specified in the Server section of the configuration.

  Type: string

  Required:

- **Level**

  Supported values are ERROR | WARNING | NOTICE |INFO | DEBUG.

  Type: string

  Required:

Warning

The DEBUG log level is unsafe for production use.

Parameters section

The Parameters section holds the network parameters, for example:

```
[Parameters]
    SendRatePerMinute = 0
    Mu = 0.005
```

```
MuMaxDelay = 1000
LambdaP = 0.001
LambdaPMaxDelay = 1000
LambdaL = 0.0005
LambdaLMaxDelay = 1000
LambdaD = 0.0005
LambdaDMaxDelay = 3000
LambdaM = 0.0005
LambdaG = 0.0
LambdaMMaxDelay = 100
LambdaGMaxDelay = 100
```

- **SendRatePerMinute**

  SendRatePerMinute is the maximum allowed rate of packets per client per gateway node. Rate limiting is done on the gateway nodes.

  SendRatePerMinute is the rate per minute.

  Type: uint64

  Required:

- **Mu**

  Mu is the inverse of the mean of the exponential distribution that the Sphinx packet per-hop mixing delay will be sampled from. Mu is the inverse of the mean of the exponential distribution that is used to select the delay for each hop.

  Type: float64

  Required:

- **MuMaxDelay**

  MuMaxDelay is the maximum Sphinx packet per-hop mixing delay in milliseconds. MuMaxDelay sets the maximum delay for Mu.

  Type: uint64

  Required:

- **LambdaP**

  LambdaP is the inverse of the mean of the exponential distribution that clients will sample to determine the time interval between sending messages from it's FIFO egress queue or drop decoy messages if the queue is empty. LambdaP is the inverse of the mean of the exponential distribution that is used to select the delay between clients sending from their egress FIFO queue or drop decoy message.

  Type: float64

  Required:

- **LambdaPMaxDelay**

  LambdaPMaxDelay is the maximum send interval for LambdaP in milliseconds

  LambdaPMaxDelay sets the maximum delay for LambdaP.

Type: uint64

Required:

- **LambdaL**

  LambdaL is the inverse of the mean of the exponential distribution that is used to select the delay between clients sending loop decoys.

  Type: float64

  Required:

- **LambdaLMaxDelay**

  sets the maximum send interval for LambdaL in milliseconds.

  LambdaLMaxDelay sets the maximum delay for LambdaP.

  Type: uint64

  Required:

- **LambdaD**

  LambdaD is the inverse of the mean of the exponential distribution that clients will sample to determine the time interval between sending decoy drop messages. LambdaD is the inverse of the mean of the exponential distribution that is used to select the delay between clients sending deop decoys.

  Type: float64

  Required:

- **LambdaDMaxDelay**

  LambdaDMaxDelay is the maximum send interval in milliseconds. LambdaDMaxDelay sets the maximum delay for LambdaP.

  Type: uint64

  Required:

- **LambdaM**

  LambdaM is the inverse of the mean of the exponential distribution that mixes will sample to determine send timing of mix loop decoy traffic. LambdaM is the inverse of the mean of the exponential distribution that is used to select the delay between sending mix node decoys.

  Type: float64

  Required:

- **LambdaG**

  LambdaG is the inverse of the mean of the exponential distribution that is used to select the delay between sending gateway node decoys. WARNING: DO NOT SET THIS VALUE. This is not used via the TOML config file; this field is used internally by the dirauth server state machine.

Type: float64

Required:

- **LambdaMMaxDelay**

  LambdaMMaxDelay sets the maximum delay for LambdaM LambdaMMaxDelay sets the maximum delay for LambdaP.

  Type: uint64

  Required:

- **LambdaGMaxDelay**

  LambdaGMaxDelay sets the maximum delay for LambdaG.

  Type: uint64

  Required:

Debug section

```
[Debug]
    Layers = 3
    MinNodesPerLayer = 1
    GenerateOnly = false
```

- **Layers**

  Number of non-provider layers in the network topology.

  Type: int

  Required:

- **MinNodesrPerLayer**

  Minimum number of nodes per layer required to form a valid document.

  Type: int

  Required:

- **GenerateOnly**

  If set to true, the server halts and cleans up the data directory immediately after long-term key generation.

  Type: bool

  Required:

Mixes sections

The Mixes configuration section lists mix nodes that are known to the authority.

```
[[Mixes]]
    Identifier = "mix1"
    IdentityPublicKeyPem = "../mix1/identity.public.pem"

[[Mixes]]
    Identifier = "mix2"
    IdentityPublicKeyPem = "../mix2/identity.public.pem"

[[Mixes]]
    Identifier = "mix3"
    IdentityPublicKeyPem = "../mix3/identity.public.pem"
```

- **Identifier**

  A human readable mix node identifier.

  Type: string

  Required:

- **IdentityPublicKeyPem**

  Path and file name of a mix node's public identity signing key, also known as the identity key, in PEM format.

  Type: string

  Required:

GatewayNodes sections

The GatewayNodes configuration section lists gateway nodes that are known to the authority.

```
[[GatewayNodes]]
    Identifier = "gateway1"
    IdentityPublicKeyPem = "../gateway1/identity.public.pem"
```

- **Identifier**

  A human readable gateway node identifier.

  Type: string

  Required:

- **IdentityPublicKeyPem**

  Path and file name of a gateway node's public identity signing key, also known as the identity key, in PEM format.

  Type: string

  Required:

ServiceNodes sections

The ServiceNodes configuration section lists service nodes that are known to the authority.

```
[[ServiceNodes]]
    Identifier = "servicenode1"
    IdentityPublicKeyPem = "../servicenode1/identity.public.pem"
```

- **Identifier**

  A human readable service node identifier.

  Type: string

  Required:

- **IdentityPublicKeyPem**

  Path and file name of a service node's public identity signing key, also known as the identity key, in PEM format.

  Type: string

  Required:

Topology section

The Topology configuration section defines the layers of the mix network and the mix nodes in each layer.

```
[Topology]

    [[Topology.Layers]]

        [[Topology.Layers.Nodes]]
            Identifier = "mix1"
            IdentityPublicKeyPem = "../mix1/identity.public.pem"

    [[Topology.Layers]]

        [[Topology.Layers.Nodes]]
            Identifier = "mix2"
            IdentityPublicKeyPem = "../mix2/identity.public.pem"

    [[Topology.Layers]]

        [[Topology.Layers.Nodes]]
            Identifier = "mix3"
            IdentityPublicKeyPem = "../mix3/identity.public.pem"
```

- **Identifier**

  A human readable mix node identifier.

  Type: string

- **IdentityPublicKeyPem**

  Path and file name of a mix node's public identity signing key, also known as the identity key, in PEM format.

  Type: string

Required:

SphinxGeometry section

WARNING: The Sphinx Geometry section of the configuration MUST be programmatically generated. Do NOT modify the geometry by hand.

The original Sphinx paper introduces the Sphinx nested encrypted packet format using a NIKE (NIKE: non-interactive key exchange). NIKE Sphinx can be a hybrid post quantum construction simply by using a hybrid NIKE. Our Sphinx implementation also can optionally use a KEM (KEM: key encapsulation mechanism) instead of a NIKE, however the trade-off is that the packet's header will take up a lot of overhead because it must store a KEM ciphertext for each hop. EchoMix has a completely configurable Sphinx geometry which allows for any KEM or NIKE to be used.

The Sphinx cryptographic packet format also uses these additional cryptographic primitives, the current EchoMix selection is:

- stream cipher: CTR-AES256
- MAC: HMAC-SHA256
- KDF: HKDF-SHA256
- SPRP: AEZv5

In EchoMix the dirauths select the Sphinx geometry, each dirauth must agree with the other dirauths. They publish the hash of the Sphinx Geometry in the PKI document so that the rest of the network entities can validate their Sphinx Geometry.

```
[SphinxGeometry]
    PacketLength = 3082
    NrHops = 5
    HeaderLength = 476
    RoutingInfoLength = 410
    PerHopRoutingInfoLength = 82
    SURBLength = 572
    SphinxPlaintextHeaderLength = 2
    PayloadTagLength = 32
    ForwardPayloadLength = 2574
    UserForwardPayloadLength = 2000
    NextNodeHopLength = 65
    SPRPKeyMaterialLength = 64
    NIKEName = "x25519"
    KEMName = ""
```

- **PacketLength**

  PacketLength is the length of a packet.

  Type: int

  Required:

- **NrHops**

  NrHops is the number of hops through the mixnet, which determines the size

of the Sphinx packet header.

Type: int

Required:

- **HeaderLength**

  HeaderLength is the length of the Sphinx packet header in bytes.

  Type: int

  Required:

- **RoutingInfoLength**

  RoutingInfoLength is the length of the routing info portion of the header.

  Type: int

  Required:

- **PerHopRoutingInfoLength**

  PerHopRoutingInfoLength is the length of the per hop routing info.

  Type: int

  Required:

- **SURBLength**

  SURBLength is the length of SURB.

  Type: int

  Required:

- **SphinxPlaintextHeaderLength**

  SphinxPlaintextHeaderLength is the length of the plaintext header.

  Type: int

  Required:

- **PayloadTagLength**

  PayloadTagLength is the length of the payload tag.

  Type: int

  Required:

- **ForwardPayloadLength**

  ForwardPayloadLength is the size of the payload.

  Type: int

Required:

- **UserForwardPayloadLength**

  the size of the usable payload.

  Type: int

  Required:

- **NextNodeHopLength**

  NextNodeHopLength is derived off the largest routing info

  block that we expect to encounter. Everything else just has a NextNodeHop + NodeDelay, or a Recipient, both cases which are shorter.

  Type: int

  Required:

- **SPRPKeyMaterialLength**

  SPRPKeyMaterialLength is the length of the SPRP key.

  Type: int

  Required:

- **NIKEName**

  NIKEName is the name of the NIKE (non-interactive key exchange) scheme used by the mixnet's Sphinx packets. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

- **KEMName**

  KEMName is the name of the KEM scheme used by the mixnet's Sphinx packet. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

Mix, gateway, and service nodes

Configuring mix nodes

The following configuration is drawn from the reference implementation in `katzenpost/docker/voting_mixnet/mix1/katzenpost.toml`. In a real-world mixnet, the component hosts would not be sharing a single IP address. For more information about the test mixnet, see Using the EchoMix test network.

Server section

```
[Server]
  Identifier = "mix1"
  WireKEM = "xwing"
  PKISignatureScheme = "Ed25519"
  Addresses = ["127.0.0.1:30008"]
  OnlyAdvertiseAltAddresses = false
  MetricsAddress = "127.0.0.1:30009"
  DataDir = "/voting_mixnet/mix1"
  IsGatewayNode = false
  IsServiceNode = false
  [Server.AltAddresses]
```

- **Identifier**

  Identifier is the human-readable identifier for a node (for example, the FQDN) and must be unique for a given mixnet.

  Type: string

  Required: Yes

- **WireKEM**

  WireKEM is the wire protocol KEM (key encapsulation mechanism) scheme to use.

  This configuration option supports the use of post-quantum cryptography to strengthen security. In this case the "Wire" protocol refers to our PQ Noise based protocol which is how all the dir auth nodes talk to one another. PQ Noise is a variation of the Noise protocol framework where the handshake patterns are algebraically transformed to replace ECDH operations with KEM encapsulate/decapsulate operations.

  The following KEM schemes are supported: "x25519", "x448", "mlkem768", "sntrup4591761", "frodo640shake", "mceliece348864", "mceliece348864f", "mceliece460896", "mceliece460896f", "mceliece6688128", "mceliece6688128f", "mceliece6960119", "mceliece6960119f", "mceliece8192128", "mceliece8192128f", "xwing", "Kyber768-X25519", "MLKEM768-X25519", "MLKEM768-X448", "CTIDH511", "CTIDH512", "CTIDH1024", "CTIDH2048", "CTIDH512-X25519", "CTIDH512-X25519"

  Type: string

  Required: Yes

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme used by all components of the mix network when interacting with the PKI system. Mix nodes sign their descriptors using this signature scheme. Likewise, PKI documents are signed by directory authority nodes using this signature scheme.

  The following signature schemes are supported: "ed25519", "ed448", "Ed25519 Sphincs+", "Ed448-Sphincs+", "Ed25519-Dilithium2", "Ed448-Dilithium3"

  Type: string

  Required: Yes

- **Addresses**

  Addresses specifies a list of one or more address URLs in a format that includes the transport protocol, IP address, and port number that the server will bind to for incoming connections. Currently EchoMix supports URLs with that start with either "tcp://" or "quic://" such as: ["tcp://192.168.1.1:30001"] and ["quic://192.168.1.1:40001"]

  Type: []string

  Required:

- **BindAddresses**

  BindAddresses are listener addresses that the server will bind to and accept connections on. These addresses are not advertised in the PKI.

  Type: bool

  Required:

- **MetricsAddress**

  MetricsAddress is the address/port to bind the Prometheus metrics endpoint to.

  Type: string

  Required:

- **DataDir**

  DataDir is the absolute path to the directory authority server state directory where persistence.db will be written to disk and where the servers' cryptographic key materials will be written to disk when the serrver is started with the "-g" commmand-line option.

  Type: string

  Required:

- **IsGatewayNode**

  IsGatewayNode specifies if the server is a gateway or not.

  Type: bool

  Required:

- **IsServiceNode**

  IsServiceNode specifies if the server is a service node or not.

  Type: bool

  Required:

Logging section

The logging configuration section controls logging.

```
[Logging]
    Disable = false
    File = "katzenpost.log"
    Level = "INFO"
```

- **Disable**

  Disables logging if set to **true**.

  Type: bool

  Required:

- **File**

  Specifies the log file. If omitted, stdout is used.

  An absolute or relative file path can be specified. A relative path is relative to the DataDir specified in the Server section of the configuration.

  Type: string

  Required:

- **Level**

  Supported values are ERROR | WARNING | NOTICE |INFO | DEBUG.

  Type: string

  Required:

Warning

The DEBUG log level is unsafe for production use.

PKI section

The PKI section contains the directory authority configuration for a mix, gateway, or service node.

```
[PKI]
    [PKI.Voting]

        [[PKI.Voting.Authorities]]
            Identifier = "auth1"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\n/v3qYgh2TvV5Z
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nJeFaZoYQEOO71zPFFWj
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30001"]

        [[PKI.Voting.Authorities]]
            Identifier = "auth2"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\n60KQRhG7njt+k
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nHVR2m7i6G6cf1qxUvyE
```

```
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30002"]

        [[PKI.Voting.Authorities]]
            Identifier = "auth3"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\naZUXqznyLO2mK
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nEZukXtZwHTjGj7tCI0k
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30003"]
```

- **Identifier**

  Identifier is the human readable identifier for the node (eg: FQDN).

  Type: string

  Required:

- **IdentityPublicKey**

  IdentityPublicKeyPem is a string in PEM format containing the public identity key.

  Type: string

  Required:

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme

  Type: string

  Required:

- **LinkPublicKey**

  LinkPublicKeyPem is string containing the PEM format of the peer's public link layer key.

  Type: string

  Required:

- **WireKEMScheme**

  WireKEMScheme is the wire protocol KEM scheme to use.

  Type: string

  Required:

- **Addresses**

  // Addresses are the IP address/port combinations that the peer authority

  // uses for the Directory Authority service.

  Type: []string

Required:

Management section

Management is the EchoMix management interface configuration. The management section specifies connectivity information for the EchoMix control protocol which can be used to make configuration changes during run-time. An example configuration looks like this:

```
[Management]
    Enable = false
    Path = "/voting_mixnet/mix1/management_sock"
```

- **Enable**

  Enables the management interface if set to true.

  Type: bool

  Required:

- **Path**

  Specifies the path to the management interface socket. If left empty, then management_sock will be used under the DataDir.

  Type: string

  Required:

SphinxGeometry section

```
[SphinxGeometry]
    PacketLength = 3082
    NrHops = 5
    HeaderLength = 476
    RoutingInfoLength = 410
    PerHopRoutingInfoLength = 82
    SURBLength = 572
    SphinxPlaintextHeaderLength = 2
    PayloadTagLength = 32
    ForwardPayloadLength = 2574
    UserForwardPayloadLength = 2000
    NextNodeHopLength = 65
    SPRPKeyMaterialLength = 64
    NIKEName = "x25519"
    KEMName = ""
```

- **PacketLength**

  PacketLength is the length of a packet.

  Type: int

  Required:

- **NrHops**

  NrHops is the number of hops through the mixnet, which determines the size

  of the Sphinx packet header.

  Type: int

  Required:

- **HeaderLength**

  HeaderLength is the length of the Sphinx packet header in bytes.

  Type: int

  Required:

- **RoutingInfoLength**

  RoutingInfoLength is the length of the routing info portion of the header.

  Type: int

  Required:

- **PerHopRoutingInfoLength**

  PerHopRoutingInfoLength is the length of the per hop routing info.

  Type: int

  Required:

- **SURBLength**

  SURBLength is the length of SURB.

  Type: int

  Required:

- **SphinxPlaintextHeaderLength**

  SphinxPlaintextHeaderLength is the length of the plaintext header.

  Type: int

  Required:

- **PayloadTagLength**

  PayloadTagLength is the length of the payload tag.

  Type: int

  Required:

- **ForwardPayloadLength**

ForwardPayloadLength is the size of the payload.

Type: int

Required:

- **UserForwardPayloadLength**

  the size of the usable payload.

  Type: int

  Required:

- **NextNodeHopLength**

  NextNodeHopLength is derived off the largest routing info

  block that we expect to encounter. Everything else just has a NextNodeHop + NodeDelay, or a Recipient, both cases which are shorter.

  Type: int

  Required:

- **SPRPKeyMaterialLength**

  SPRPKeyMaterialLength is the length of the SPRP key.

  Type: int

  Required:

- **NIKEName**

  NIKEName is the name of the NIKE (non-interactive key exchange) scheme used by the mixnet's Sphinx packets. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

- **KEMName**

  KEMName is the name of the KEM scheme used by the mixnet's Sphinx packet. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

Debug section

Debug is the EchoMix server debug configuration for advanced tuning.

```
[Debug]
```

```
NumSphinxWorkers = 16
NumServiceWorkers = 3
NumGatewayWorkers = 3
NumKaetzchenWorkers = 3
SchedulerExternalMemoryQueue = false
SchedulerQueueSize = 0
SchedulerMaxBurst = 16
UnwrapDelay = 250
GatewayDelay = 500
ServiceDelay = 500
KaetzchenDelay = 750
SchedulerSlack = 150
SendSlack = 50
DecoySlack = 15000
ConnectTimeout = 60000
HandshakeTimeout = 30000
ReauthInterval = 30000
SendDecoyTraffic = false
DisableRateLimit = false
GenerateOnly = false
```

- **NumSphinxWorkers**

  specifies the number of worker instances to use for inbound Sphinx packet processing.

  Type: int

  Required:

- **NumProviderWorkers**

  specifies the number of worker instances to use for provider specific packet processing.

  Type: int

  Required:

- **NumKaetzchenWorkers**

  specifies the number of worker instances to use for Kaetzchen specific packet processing.

  Type: int

  Required:

- **SchedulerExternalMemoryQueue**

  will enable the experimental external memory queue that is backed by disk.

  Type: bool

  Required:

- **SchedulerQueueSize**

  is the maximum allowed scheduler queue size before random entries will start getting dropped. A value <= 0 is treated as unlimited.

Type: int

Required:

- **SchedulerMaxBurst**

  is the maximum number of packets that will be dispatched per scheduler wakeup event.

  Type:

  Required:

- **UnwrapDelay**

  is the maximum allowed unwrap delay due to queueing in milliseconds.

  Type: int

  Required:

- **GatewayDelay**

  the maximum allowed gateway node worker delay due to queueing

  in milliseconds.

  Type: int

  Required:

- **ServiceDelay**

  is the maximum allowed provider delay due to queueing in milliseconds.

  Type: int

  Required:

- **KaetzchenDelay**

  is the maximum allowed kaetzchen delay due to queueing in milliseconds.

  Type: int

  Required:

- **SchedulerSlack**

  is the maximum allowed scheduler slack due to queueing and or processing in milliseconds.

  Type: int

  Required:

- **SendSlack**

  is the maximum allowed send queue slack due to queueing and or congestion in milliseconds.

Type: int

Required:

- **DecoySlack**

  is the maximum allowed decoy sweep slack due to various external delays such as latency before a loop decoy packet will be considered lost.

  Type: int

  Required:

- **ConnectTimeout**

  specifies the maximum time a connection can take to establish a TCP/IP connection in milliseconds.

  Type: int

  Required:

- **HandshakeTimeout**

  specifies the maximum time a connection can take for a link protocol handshake in milliseconds.

  Type: int

  Required:

- **ReauthInterval**

  specifies the interval at which a connection will be reauthenticated in milliseconds.

  Type: int

  Required:

- **SendDecoyTraffic**

  enables sending decoy traffic. This is still experimental and untuned and thus is disabled by default. WARNING: This option will go away once decoy traffic is more concrete.

  Type: bool

  Required:

- **DisableRateLimit**

  disables the per-client rate limiter. This option should only be used for testing.

  Type: bool

  Required:

- **GenerateOnly**

  halts and cleans up the server right after long term key generation.

Type: bool

Required:

Configuring gateway nodes

The following configuration is drawn from the reference implementation in `katzenpost/docker/voting_mixnet/gateway1/katzenpost.toml`. In a real-world mixnet, the component hosts would not be sharing a single IP address. For more information about the test mixnet, see Using the EchoMix test network.

Server section

```
[Server]
    Identifier = "gateway1"
    WireKEM = "xwing"
    PKISignatureScheme = "Ed25519"
    Addresses = ["127.0.0.1:30004"]
    OnlyAdvertiseAltAddresses = false
    MetricsAddress = "127.0.0.1:30005"
    DataDir = "/voting_mixnet/gateway1"
    IsGatewayNode = true
    IsServiceNode = false
    [Server.AltAddresses]
        TCP = ["localhost:30004"]
```

- **Identifier**

  Identifier is the human-readable identifier for a node (for example, the FQDN) and must be unique for a given mixnet.

  Type: string

  Required: Yes

- **WireKEM**

  WireKEM is the wire protocol KEM (key encapsulation mechanism) scheme to use.

  This configuration option supports the use of post-quantum cryptography to strengthen security. In this case the "Wire" protocol refers to our PQ Noise based protocol which is how all the dir auth nodes talk to one another. PQ Noise is a variation of the Noise protocol framework where the handshake patterns are algebraically transformed to replace ECDH operations with KEM encapsulate/decapsulate operations.

  The following KEM schemes are supported: "x25519", "x448", "mlkem768", "sntrup4591761", "frodo640shake", "mceliece348864", "mceliece348864f", "mceliece460896", "mceliece460896f", "mceliece6688128", "mceliece6688128f", "mceliece6960119", "mceliece6960119f", "mceliece8192128", "mceliece8192128f", "xwing", "Kyber768-X25519", "MLKEM768-X25519", "MLKEM768-X448", "CTIDH511", "CTIDH512", "CTIDH1024", "CTIDH2048", "CTIDH512-X25519", "CTIDH512-X25519"

  Type: string

  Required: Yes

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme used by all components of the mix network when interacting with the PKI system. Mix nodes sign their descriptors using this signature scheme. Likewise, PKI documents are signed by directory authority nodes using this signature scheme.

  The following signature schemes are supported: "ed25519", "ed448", "Ed25519 Sphincs+", "Ed448-Sphincs+", "Ed25519-Dilithium2", "Ed448-Dilithium3"

  Type: string

  Required: Yes

- **Addresses**

  Addresses specifies a list of one or more address URLs in a format that includes the transport protocol, IP address, and port number that the server will bind to for incoming connections. Currently EchoMix supports URLs with that start with either "tcp://" or "quic://" such as: ["tcp://192.168.1.1:30001"] and ["quic://192.168.1.1:40001"]

  Type: []string

  Required:

- **BindAddresses**

  BindAddresses are listener addresses that the server will bind to and accept connections on. These addresses are not advertised in the PKI.

  Type: bool

  Required:

- **MetricsAddress**

  MetricsAddress is the address/port to bind the Prometheus metrics endpoint to.

  Type: string

  Required:

- **DataDir**

  DataDir is the absolute path to the directory authority server state directory where persistence.db will be written to disk and where the servers' cryptographic key materials will be written to disk when the serrver is started with the "-g" commmand-line option.

  Type: string

  Required:

- **IsGatewayNode**

  IsGatewayNode specifies if the server is a gateway or not.

  Type: bool

  Required:

- **IsServiceNode**

  IsServiceNode specifies if the server is a service node or not.

  Type: bool

  Required:

Logging section

The logging configuration section controls logging.

```
[Logging]
    Disable = false
    File = "katzenpost.log"
    Level = "INFO"
```

- **Disable**

  Disables logging if set to **true**.

  Type: bool

  Required:

- **File**

  Specifies the log file. If omitted, stdout is used.

  An absolute or relative file path can be specified. A relative path is relative to the DataDir specified in the Server section of the configuration.

  Type: string

  Required:

- **Level**

  Supported values are ERROR | WARNING | NOTICE |INFO | DEBUG.

  Type: string

  Required:

Warning

The DEBUG log level is unsafe for production use.

Gateway section

```
[Gateway]
    [Gateway.UserDB]
        Backend = "bolt"
            [Gateway.UserDB.Bolt]
                UserDB = "/voting_mixnet/gateway1/users.db"
    [Gateway.SpoolDB]
```

```
        Backend = "bolt"
            [Gateway.SpoolDB.Bolt]
                SpoolDB = "/voting_mixnet/gateway1/spool.db"
```

-
-
-
-
-
-
-
-
-
-
-

PKI section

The PKI section contains the directory authority configuration for a mix, gateway, or service node.

```
[PKI]
    [PKI.Voting]

        [[PKI.Voting.Authorities]]
            Identifier = "auth1"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\n/v3qYgh2TvV5Z
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nJeFaZoYQEOO71zPFFWj
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30001"]

        [[PKI.Voting.Authorities]]
            Identifier = "auth2"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\n60KQRhG7njt+k
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nHVR2m7i6G6cf1qxUvyE
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30002"]

        [[PKI.Voting.Authorities]]
            Identifier = "auth3"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\naZUXqznyLO2mK
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nEZukXtZwHTjGj7tCI0k
```

```
WireKEMScheme = "xwing"
Addresses = ["127.0.0.1:30003"]
```

- **Identifier**

  Identifier is the human readable identifier for the node (eg: FQDN).

  Type: string

  Required:

- **IdentityPublicKey**

  IdentityPublicKeyPem is a string in PEM format containing the public identity key.

  Type: string

  Required:

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme

  Type: string

  Required:

- **LinkPublicKey**

  LinkPublicKeyPem is string containing the PEM format of the peer's public link layer key.

  Type: string

  Required:

- **WireKEMScheme**

  WireKEMScheme is the wire protocol KEM scheme to use.

  Type: string

  Required:

- **Addresses**

  // Addresses are the IP address/port combinations that the peer authority

  // uses for the Directory Authority service.

  Type: []string

  Required:

Management section

Management is the EchoMix management interface configuration. The management section specifies connectivity information for the EchoMix control protocol which can be used to make configuration changes during run-time. An example configuration looks like this:

```
[Management]
    Enable = false
    Path = "/voting_mixnet/mix1/management_sock"
```

- **Enable**

  Enables the management interface if set to true.

  Type: bool

  Required:

- **Path**

  Specifies the path to the management interface socket. If left empty, then management_sock will be used under the DataDir.

  Type: string

  Required:

SphinxGeometry section

```
[SphinxGeometry]
    PacketLength = 3082
    NrHops = 5
    HeaderLength = 476
    RoutingInfoLength = 410
    PerHopRoutingInfoLength = 82
    SURBLength = 572
    SphinxPlaintextHeaderLength = 2
    PayloadTagLength = 32
    ForwardPayloadLength = 2574
    UserForwardPayloadLength = 2000
    NextNodeHopLength = 65
    SPRPKeyMaterialLength = 64
    NIKEName = "x25519"
    KEMName = ""
```

- **PacketLength**

  PacketLength is the length of a packet.

  Type: int

  Required:

- **NrHops**

  NrHops is the number of hops through the mixnet, which determines the size

  of the Sphinx packet header.

  Type: int

  Required:

- **HeaderLength**

  HeaderLength is the length of the Sphinx packet header in bytes.

  Type: int

  Required:

- **RoutingInfoLength**

  RoutingInfoLength is the length of the routing info portion of the header.

  Type: int

  Required:

- **PerHopRoutingInfoLength**

  PerHopRoutingInfoLength is the length of the per hop routing info.

  Type: int

  Required:

- **SURBLength**

  SURBLength is the length of SURB.

  Type: int

  Required:

- **SphinxPlaintextHeaderLength**

  SphinxPlaintextHeaderLength is the length of the plaintext header.

  Type: int

  Required:

- **PayloadTagLength**

  PayloadTagLength is the length of the payload tag.

  Type: int

  Required:

- **ForwardPayloadLength**

  ForwardPayloadLength is the size of the payload.

  Type: int

  Required:

- **UserForwardPayloadLength**

  the size of the usable payload.

Type: int

Required:

- **NextNodeHopLength**

  NextNodeHopLength is derived off the largest routing info

  block that we expect to encounter. Everything else just has a NextNodeHop + NodeDelay, or a Recipient, both cases which are shorter.

  Type: int

  Required:

- **SPRPKeyMaterialLength**

  SPRPKeyMaterialLength is the length of the SPRP key.

  Type: int

  Required:

- **NIKEName**

  NIKEName is the name of the NIKE (non-interactive key exchange) scheme used by the mixnet's Sphinx packets. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

- **KEMName**

  KEMName is the name of the KEM scheme used by the mixnet's Sphinx packet. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

Debug section

Debug is the EchoMix server debug configuration for advanced tuning.

```
[Debug]
                    NumSphinxWorkers = 16
                    NumServiceWorkers = 3
                    NumGatewayWorkers = 3
                    NumKaetzchenWorkers = 3
                    SchedulerExternalMemoryQueue = false
                    SchedulerQueueSize = 0
                    SchedulerMaxBurst = 16
                    UnwrapDelay = 250
                    GatewayDelay = 500
                    ServiceDelay = 500
```

```
KaetzchenDelay = 750
SchedulerSlack = 150
SendSlack = 50
DecoySlack = 15000
ConnectTimeout = 60000
HandshakeTimeout = 30000
ReauthInterval = 30000
SendDecoyTraffic = false
DisableRateLimit = false
GenerateOnly = false
```

- **NumSphinxWorkers**

  specifies the number of worker instances to use for inbound Sphinx packet processing.

  Type: int

  Required:

- **NumProviderWorkers**

  specifies the number of worker instances to use for provider specific packet processing.

  Type: int

  Required:

- **NumKaetzchenWorkers**

  specifies the number of worker instances to use for Kaetzchen specific packet processing.

  Type: int

  Required:

- **SchedulerExternalMemoryQueue**

  will enable the experimental external memory queue that is backed by disk.

  Type: bool

  Required:

- **SchedulerQueueSize**

  is the maximum allowed scheduler queue size before random entries will start getting dropped. A value <= 0 is treated as unlimited.

  Type: int

  Required:

- **SchedulerMaxBurst**

  is the maximum number of packets that will be dispatched per scheduler wakeup event.

  Type:

Required:

- **UnwrapDelay**

  is the maximum allowed unwrap delay due to queueing in milliseconds.

  Type: int

  Required:

- **GatewayDelay**

  the maximum allowed gateway node worker delay due to queueing

  in milliseconds.

  Type: int

  Required:

- **ServiceDelay**

  is the maximum allowed provider delay due to queueing in milliseconds.

  Type: int

  Required:

- **KaetzchenDelay**

  is the maximum allowed kaetzchen delay due to queueing in milliseconds.

  Type: int

  Required:

- **SchedulerSlack**

  is the maximum allowed scheduler slack due to queueing and or processing in milliseconds.

  Type: int

  Required:

- **SendSlack**

  is the maximum allowed send queue slack due to queueing and or congestion in milliseconds.

  Type: int

  Required:

- **DecoySlack**

  is the maximum allowed decoy sweep slack due to various external delays such as latency before a loop decoy packet will be considered lost.

  Type: int

Required:

- **ConnectTimeout**

  specifies the maximum time a connection can take to establish a TCP/IP connection in milliseconds.

  Type: int

  Required:

- **HandshakeTimeout**

  specifies the maximum time a connection can take for a link protocol handshake in milliseconds.

  Type: int

  Required:

- **ReauthInterval**

  specifies the interval at which a connection will be reauthenticated in milliseconds.

  Type: int

  Required:

- **SendDecoyTraffic**

  enables sending decoy traffic. This is still experimental and untuned and thus is disabled by default. WARNING: This option will go away once decoy traffic is more concrete.

  Type: bool

  Required:

- **DisableRateLimit**

  disables the per-client rate limiter. This option should only be used for testing.

  Type: bool

  Required:

- **GenerateOnly**

  halts and cleans up the server right after long term key generation.

  Type: bool

  Required:

Configuring service nodes

The following configuration is drawn from the reference implementation in `katzenpost/docker/voting_mixnet/servicenode1/authority.toml`. In a real-world mixnet, the component hosts would not be sharing a single IP address. For more information about the test mixnet, see Using the Katzenpost test network.

Server section

The Server section contains mandatory information common to all nodes, for example:

```
[Server]
    Identifier = "servicenode1"
    WireKEM = "xwing"
    PKISignatureScheme = "Ed25519"
    Addresses = ["127.0.0.1:30006"]
    OnlyAdvertiseAltAddresses = false
    MetricsAddress = "127.0.0.1:30007"
    DataDir = "/voting_mixnet/servicenode1"
    IsGatewayNode = false
    IsServiceNode = true
    [Server.AltAddresses]
```

- **Identifier**

  Identifier is the human-readable identifier for a node (for example, the FQDN) and must be unique for a given mixnet.

  Type: string

  Required: Yes

- **WireKEM**

  WireKEM is the wire protocol KEM (key encapsulation mechanism) scheme to use.

  This configuration option supports the use of post-quantum cryptography to strengthen security. In this case the "Wire" protocol refers to our PQ Noise based protocol which is how all the dir auth nodes talk to one another. PQ Noise is a variation of the Noise protocol framework where the handshake patterns are algebraically transformed to replace ECDH operations with KEM encapsulate/decapsulate operations.

  The following KEM schemes are supported: "x25519", "x448", "mlkem768", "sntrup4591761", "frodo640shake", "mceliece348864", "mceliece348864f", "mceliece460896", "mceliece460896f", "mceliece6688128", "mceliece6688128f", "mceliece6960119", "mceliece6960119f", "mceliece8192128", "mceliece8192128f", "xwing", "Kyber768-X25519", "MLKEM768-X25519", "MLKEM768-X448", "CTIDH511", "CTIDH512", "CTIDH1024", "CTIDH2048", "CTIDH512-X25519", "CTIDH512-X25519"

  Type: string

  Required: Yes

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme used by all components of the mix network when interacting with the PKI system. Mix nodes sign their descriptors using this signature scheme. Likewise, PKI documents are signed by directory authority nodes using this signature scheme.

  The following signature schemes are supported: "ed25519", "ed448", "Ed25519 Sphincs+", "Ed448-Sphincs+", "Ed25519-Dilithium2", "Ed448-Dilithium3"

  Type: string

  Required: Yes

- **Addresses**

  Addresses specifies a list of one or more address URLs in a format that includes the transport protocol, IP address, and port number that the server will bind to for incoming connections. Currently EchoMix supports URLs with that start with either "tcp://" or "quic://" such as: ["tcp://192.168.1.1:30001"] and ["quic://192.168.1.1:40001"]

  Type: []string

  Required:

- **BindAddresses**

  BindAddresses are listener addresses that the server will bind to and accept connections on. These addresses are not advertised in the PKI.

  Type: bool

  Required:

- **MetricsAddress**

  MetricsAddress is the address/port to bind the Prometheus metrics endpoint to.

  Type: string

  Required:

- **DataDir**

  DataDir is the absolute path to the directory authority server state directory where persistence.db will be written to disk and where the servers' cryptographic key materials will be written to disk when the serrver is started with the "-g" commmand-line option.

  Type: string

  Required:

- **IsGatewayNode**

  IsGatewayNode specifies if the server is a gateway or not.

  Type: bool

  Required:

- **IsServiceNode**

  IsServiceNode specifies if the server is a service node or not.

  Type: bool

  Required:

Logging section

The logging configuration section controls logging.

```
[Logging]
    Disable = false
    File = "katzenpost.log"
    Level = "INFO"
```

- **Disable**

  Disables logging if set to **true**.

  Type: bool

  Required:

- **File**

  Specifies the log file. If omitted, stdout is used.

  An absolute or relative file path can be specified. A relative path is relative to the DataDir specified in the Server section of the configuration.

  Type: string

  Required:

- **Level**

  Supported values are ERROR | WARNING | NOTICE |INFO | DEBUG.

  Type: string

  Required:

Warning

The DEBUG log level is unsafe for production use.

ServiceNode section

The service node configuration section contains subsections with settings for each service that Katzenpost supports. In a production network, the various services would be hosted on dedicated systems.

```
[ServiceNode]

    [[ServiceNode.Kaetzchen]]
        Capability = "echo"
        Endpoint = "+echo"
        Disable = false

    [[ServiceNode.CBORPluginKaetzchen]]
        Capability = "spool"
        Endpoint = "+spool"
        Command = "/voting_mixnet/memspool.alpine"
        MaxConcurrency = 1
        Disable = false
        [ServiceNode.CBORPluginKaetzchen.Config]
            data_store = "/voting_mixnet/servicenode1/memspool.storage"
```

```
            log_dir = "/voting_mixnet/servicenode1"

    [[ServiceNode.CBORPluginKaetzchen]]
        Capability = "pigeonhole"
        Endpoint = "+pigeonhole"
        Command = "/voting_mixnet/pigeonhole.alpine"
        MaxConcurrency = 1
        Disable = false
        [ServiceNode.CBORPluginKaetzchen.Config]
            db = "/voting_mixnet/servicenode1/map.storage"
            log_dir = "/voting_mixnet/servicenode1"

    [[ServiceNode.CBORPluginKaetzchen]]
        Capability = "panda"
        Endpoint = "+panda"
        Command = "/voting_mixnet/panda_server.alpine"
        MaxConcurrency = 1
        Disable = false
        [ServiceNode.CBORPluginKaetzchen.Config]
            fileStore = "/voting_mixnet/servicenode1/panda.storage"
            log_dir = "/voting_mixnet/servicenode1"
            log_level = "INFO"

    [[ServiceNode.CBORPluginKaetzchen]]
        Capability = "http"
        Endpoint = "+http"
        Command = "/voting_mixnet/proxy_server.alpine"
        MaxConcurrency = 1
        Disable = false
        [ServiceNode.CBORPluginKaetzchen.Config]
            host = "localhost:4242"
            log_dir = "/voting_mixnet/servicenode1"
            log_level = "DEBUG"
```

Common parameters:

- **Capability**

  The capability exposed by the agent.

  Type: string

  Required:

- **Endpoint**

  Endpoint is the provider side endpoint that the agent will accept requests at. While not required by the spec, this server only supports Endpoints that are lower-case local-parts of an e-mail address.

  Type: string

  Required:

- **Command**

  Command is the full file path to the external plugin program that implements this Kaetzchen service.

Type: string

Required:

- **MaxConcurrency**

  MaxConcurrency is the number of worker goroutines to start for this service.

  Type: int

  Required:

- **Config**

  The extra per agent arguments to be passed to the agent's initialization routine.

  Type: map[string]interface{}

  Required:

- **Disable**

  disabled a configured agent.

  Type: bool

  Required:

Per-service parameters:

- **Kaetzchen**
- **spool**
  - **data_store**

    Type:

    Required:

  - **log_dir**

    Type:

    Required:
- **pigeonhole**
  - **db**

    Type:

    Required:

  - **log_dir**

Type:

Required:

- **panda**
  - **fileStore**

    Type:

    Required:
  - **log_dir**

    Type:

    Required:
  - **log_level**

    Supported values are ERROR | WARNING | NOTICE |INFO | DEBUG.

    ## Warning

    The DEBUG log level is unsafe for production use.

    Type: string

    Required:

    Required:
- **http**
  - **host**

    Type:

    Required:
  - **log_dir**

    Type:

    Required:
  - **log_level**

    Supported values are ERROR | WARNING | NOTICE |INFO | DEBUG.

    Type: string

    Required:

Required:

Warning

The DEBUG log level is unsafe for production use.

Type: string

Required:

Required:

PKI section

The PKI section contains the directory authority configuration for a mix, gateway, or service node.

```
[PKI]
    [PKI.Voting]

        [[PKI.Voting.Authorities]]
            Identifier = "auth1"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\n/v3qYgh2TvV5Z
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nJeFaZoYQEOO71zPFFWj
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30001"]

        [[PKI.Voting.Authorities]]
            Identifier = "auth2"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\n60KQRhG7njt+k
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nHVR2m7i6G6cf1qxUvyE
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30002"]

        [[PKI.Voting.Authorities]]
            Identifier = "auth3"
            IdentityPublicKey = "-----BEGIN ED25519 PUBLIC KEY-----\naZUXqznyLO2mK
            PKISignatureScheme = "Ed25519"
            LinkPublicKey = "-----BEGIN XWING PUBLIC KEY-----\nEZukXtZwHTjGj7tCI0k
            WireKEMScheme = "xwing"
            Addresses = ["127.0.0.1:30003"]
```

- **Identifier**

  Identifier is the human readable identifier for the node (eg: FQDN).

  Type: string

  Required:

- **IdentityPublicKey**

  IdentityPublicKeyPem is a string in PEM format containing the public identity key.

Type: string

Required:

- **PKISignatureScheme**

  PKISignatureScheme specifies the cryptographic signature scheme

  Type: string

  Required:

- **LinkPublicKey**

  LinkPublicKeyPem is string containing the PEM format of the peer's public link layer key.

  Type: string

  Required:

- **WireKEMScheme**

  WireKEMScheme is the wire protocol KEM scheme to use.

  Type: string

  Required:

- **Addresses**

  // Addresses are the IP address/port combinations that the peer authority

  // uses for the Directory Authority service.

  Type: []string

  Required:

Management section

Management is the EchoMix management interface configuration. The management section specifies connectivity information for the EchoMix control protocol which can be used to make configuration changes during run-time. An example configuration looks like this:

```
[Management]
    Enable = false
    Path = "/voting_mixnet/mix1/management_sock"
```

- **Enable**

  Enables the management interface if set to true.

  Type: bool

  Required:

- **Path**

Specifies the path to the management interface socket. If left empty, then management_sock will be used under the DataDir.

Type: string

Required:

SphinxGeometry section

```
[SphinxGeometry]
    PacketLength = 3082
    NrHops = 5
    HeaderLength = 476
    RoutingInfoLength = 410
    PerHopRoutingInfoLength = 82
    SURBLength = 572
    SphinxPlaintextHeaderLength = 2
    PayloadTagLength = 32
    ForwardPayloadLength = 2574
    UserForwardPayloadLength = 2000
    NextNodeHopLength = 65
    SPRPKeyMaterialLength = 64
    NIKEName = "x25519"
    KEMName = ""
```

- **PacketLength**

  PacketLength is the length of a packet.

  Type: int

  Required:

- **NrHops**

  NrHops is the number of hops through the mixnet, which determines the size

  of the Sphinx packet header.

  Type: int

  Required:

- **HeaderLength**

  HeaderLength is the length of the Sphinx packet header in bytes.

  Type: int

  Required:

- **RoutingInfoLength**

  RoutingInfoLength is the length of the routing info portion of the header.

  Type: int

Required:

- **PerHopRoutingInfoLength**

  PerHopRoutingInfoLength is the length of the per hop routing info.

  Type: int

  Required:

- **SURBLength**

  SURBLength is the length of SURB.

  Type: int

  Required:

- **SphinxPlaintextHeaderLength**

  SphinxPlaintextHeaderLength is the length of the plaintext header.

  Type: int

  Required:

- **PayloadTagLength**

  PayloadTagLength is the length of the payload tag.

  Type: int

  Required:

- **ForwardPayloadLength**

  ForwardPayloadLength is the size of the payload.

  Type: int

  Required:

- **UserForwardPayloadLength**

  the size of the usable payload.

  Type: int

  Required:

- **NextNodeHopLength**

  NextNodeHopLength is derived off the largest routing info

  block that we expect to encounter. Everything else just has a NextNodeHop + NodeDelay, or a Recipient, both cases which are shorter.

  Type: int

Required:

- **SPRPKeyMaterialLength**

  SPRPKeyMaterialLength is the length of the SPRP key.

  Type: int

  Required:

- **NIKEName**

  NIKEName is the name of the NIKE (non-interactive key exchange) scheme used by the mixnet's Sphinx packets. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

- **KEMName**

  KEMName is the name of the KEM scheme used by the mixnet's Sphinx packet. NIKEName and KEMName are mutually exclusive.

  Type: string

  Required:

Debug section

Debug is the EchoMix server debug configuration for advanced tuning.

```
[Debug]
                    NumSphinxWorkers = 16
                    NumServiceWorkers = 3
                    NumGatewayWorkers = 3
                    NumKaetzchenWorkers = 3
                    SchedulerExternalMemoryQueue = false
                    SchedulerQueueSize = 0
                    SchedulerMaxBurst = 16
                    UnwrapDelay = 250
                    GatewayDelay = 500
                    ServiceDelay = 500
                    KaetzchenDelay = 750
                    SchedulerSlack = 150
                    SendSlack = 50
                    DecoySlack = 15000
                    ConnectTimeout = 60000
                    HandshakeTimeout = 30000
                    ReauthInterval = 30000
                    SendDecoyTraffic = false
                    DisableRateLimit = false
                    GenerateOnly = false
```

- **NumSphinxWorkers**

specifies the number of worker instances to use for inbound Sphinx packet processing.

Type: int

Required:

- **NumProviderWorkers**

  specifies the number of worker instances to use for provider specific packet processing.

  Type: int

  Required:

- **NumKaetzchenWorkers**

  specifies the number of worker instances to use for Kaetzchen specific packet processing.

  Type: int

  Required:

- **SchedulerExternalMemoryQueue**

  will enable the experimental external memory queue that is backed by disk.

  Type: bool

  Required:

- **SchedulerQueueSize**

  is the maximum allowed scheduler queue size before random entries will start getting dropped. A value <= 0 is treated as unlimited.

  Type: int

  Required:

- **SchedulerMaxBurst**

  is the maximum number of packets that will be dispatched per scheduler wakeup event.

  Type:

  Required:

- **UnwrapDelay**

  is the maximum allowed unwrap delay due to queueing in milliseconds.

  Type: int

  Required:

- **GatewayDelay**

  the maximum allowed gateway node worker delay due to queueing

in milliseconds.

Type: int

Required:

- **ServiceDelay**

  is the maximum allowed provider delay due to queueing in milliseconds.

  Type: int

  Required:

- **KaetzchenDelay**

  is the maximum allowed kaetzchen delay due to queueing in milliseconds.

  Type: int

  Required:

- **SchedulerSlack**

  is the maximum allowed scheduler slack due to queueing and or processing in milliseconds.

  Type: int

  Required:

- **SendSlack**

  is the maximum allowed send queue slack due to queueing and or congestion in milliseconds.

  Type: int

  Required:

- **DecoySlack**

  is the maximum allowed decoy sweep slack due to various external delays such as latency before a loop decoy packet will be considered lost.

  Type: int

  Required:

- **ConnectTimeout**

  specifies the maximum time a connection can take to establish a TCP/IP connection in milliseconds.

  Type: int

  Required:

- **HandshakeTimeout**

  specifies the maximum time a connection can take for a link protocol handshake in milliseconds.

Type: int

Required:

- **ReauthInterval**

  specifies the interval at which a connection will be reauthenticated in milliseconds.

  Type: int

  Required:

- **SendDecoyTraffic**

  enables sending decoy traffic. This is still experimental and untuned and thus is disabled by default. WARNING: This option will go away once decoy traffic is more concrete.

  Type: bool

  Required:

- **DisableRateLimit**

  disables the per-client rate limiter. This option should only be used for testing.

  Type: bool

  Required:

- **GenerateOnly**

  halts and cleans up the server right after long term key generation.

  Type: bool

  Required:

Chapter 2. Using the EchoMix Docker test network

EchoMix provides a ready-to-deploy Docker image [https://github.com/katzenpost/katzenpost/tree/main/docker] for developers who need a non-production test environment for developing and testing client applications. By running this image on a single computer, you avoid the need to build and manage a complex multi-node mix net. The image can also be run using Podman [https://podman.io/]

The test mix network includes the following components:

- Three directory authority (PKI [https://katzenpost.network/docs/specs/pki/]) nodes
- Six mix [https://katzenpost.network/docs/specs/mixnet/] nodes, including one node serving also as both gateway and service provider
- A ping utility

Requirements

Before running the EchoMix docker image, make sure that the following software is installed.

- A Debian GNU Linux [https://debian.org] or Ubuntu [https://ubuntu.com] system
- Git [https://git-scm.com/]
- Go [https://go.dev/]
- GNU Make [https://www.gnu.org/software/make/]
- Docker [https://www.docker.com], Docker Compose [https://docs.docker.com/compose/], and (optionally) Podman [https://podman.io]

Note

If both Docker and Podman are present on your system, EchoMix uses Podman. Podman is a drop-in daemonless equivalent to Docker that does not require superuser privileges to run.

On Debian, these software requirements can be installed with the following commands (running as superuser). **Apt** will pull in the needed dependencies.

```
# apt update
            # apt install git golang make docker docker-compose podman
```

Preparing to run the container image

Complete the following procedure to obtain, build, and deploy the EchoMix test network.

1. Install the EchoMix code repository, hosted at https://github.com/katzenpost. The main EchoMix repository contains code for the server components as well as the docker image. Clone the repository with the following command (your directory location may vary):

```
~$ git clone https://github.com/katzenpost/katzenpost.git
```

2. Navigate to the new `katzenpost` subdirectory and ensure that the code is up to date.

```
~$ cd katzenpost
                          ~/katzenpost$ git checkout main
                          ~/katzenpost$ git pull
```

3. (Optional) Create a development branch and check it out.

```
~/katzenpost$ git checkout -b devel
```

4. (Optional) If you are using Podman, complete the following steps:

    1. Point the DOCKER_HOST environment variable at the Podman process.

    ```
    $ export DOCKER_HOST=unix:///var/run/user/$(id -u)/podman/podman.sock
    ```

    2. Set up and start the Podman server (as superuser).

    ```
    $ podman system service -t 0 $DOCKER_HOST &
                                  $ systemctl --user enable --now podman.socket
    ```

Operating the test mixnet

Navigate to `katzenpost/docker`. The `Makefile` contains target operations to create, manage, and test the self-contained EchoMix container network. To invoke a target, run a command with the using the following pattern:

```
~/katzenpost/docker$ make target
```

Running **make** with no target specified returns a list of available targets.:

Table 2.1. Makefile targets

[none]	Display this list of targets.
run	Run the test network in the background.
start	Run the test network in the foreground until **Ctrl-C**.
stop	Stop the test network.
wait	Wait for the test network to have consensus.
watch	Display live log entries until **Ctrl-C**.
status	Show test network consensus status.
show-latest-vote	Show latest consensus vote.
run-ping	Send a ping over the test network.
clean-bin	Stop all components and delete binaries.
clean-local	Stop all components, delete binaries, and delete data..
clean-local-dryrun	Show what clean-local would delete.
clean	The above, plus cleans includes go_deps image.

Starting and monitoring the mixnet

Either of two command targets, **run** and **start**, can be used to start the mix network. The first They differ only in that **start** quickly detaches and runs the network in the background, while **run** runs the network in the foreground.

Note

When running **run** or **start** , be aware of the following considerations:

- If you intend to use Docker, you need to run **make** as superuser. If you are using **sudo** to elevate your privileges, you need to edit `katzenpost/docker/Makefile` to prepend **sudo** to each command contained in it.
- If you have Podman installed on your system and you nonetheless want to run Docker, you can override the default behavior by adding the argument **docker=docker** to the command as in the following:

  ```
  ~/katzenpost/docker$ make run docker=docker
  ```

The first time that you use the **start** target, the docker image will be downloaded, built, and installed. This takes several minutes.

```
~/katzenpost/docker$ make start
                     ...
                     ~/katzenpost/docker$ make watch
                     ...
                     <output>
                     ...
```

Once installation is complete, there is a further delay as the mix servers vote and reach a consensus. You can use the **wait** target to wait for the mixnet to get consensus and be ready to use. This can also take several minutes:

```
~/katzenpost/docker$ make wait
                     ...
                     <output>
                     ...
```

You can confirm that installation and configuration are complete by issuing the **status** command from the same or another terminal. When the network is ready for use, **status** begins returning consensus information similar to the following:

```
~/katzenpost/docker$ make status
                     ...
                     00:15:15.003 NOTI state: Consensus made for epoch 1851128 with
                     ...
```

Testing the mixnet

At this point, you should have a locally running mix network. You can test whether it is working correctly by using **ping**, which launches a packet into the network and watches for a successful reply. Run the following command:

```
~/katzenpost/docker$ make run-ping
```

If the network is functioning properly, the resulting output contains lines similar to the following:

```
19:29:53.541 INFO gateway1_client: sending loop decoy
               !19:29:54.108 INFO gateway1_client: sending loop decoy
               19:29:54.632 INFO gateway1_client: sending loop decoy
               19:29:55.160 INFO gateway1_client: sending loop decoy
               !19:29:56.071 INFO gateway1_client: sending loop decoy
               !19:29:59.173 INFO gateway1_client: sending loop decoy
               !Success rate is 100.000000 percent 10/10)
```

lf **ping** fails to receive a reply, it eventually times out with an error message. If this happens, try the command again.

Note

If you attempt use **ping** too quickly after starting the mixnet, and consensus has not been reached, the utility may crash with an error message or hang indefinitely. If this happens, issue (if necessary) a **Ctrl-C** key sequence to abort, check the consensus status with the **status** command, and then retry **ping**.

Shutting down the mixnet

The mix network continues to run in the terminal where you started it until you issue a **Ctrl-C** key sequence, or until you issue the following command in another terminal:

```
~/katzenpost/docker$ make stop
```

When you stop the network, the binaries and data are left in place. This allows for a quick restart.

Uninstalling and cleaning up

Several command targets can be used to uninstall the Docker image and restore your system to a clean state. The following examples demonstrate the commands and their output.

- **clean-bin**

  To stop the network and delete the compiled binaries, run the following command:

  ```
  ~/katzenpost/docker$ make clean-bin
  ```

  ```
  [ -e voting_mixnet ] && cd voting_mixnet && DOCKER_HOST=u
  Stopping voting_mixnet_auth3_1        ... done
  Stopping voting_mixnet_servicenode1_1 ... done
  Stopping voting_mixnet_metrics_1      ... done
  Stopping voting_mixnet_mix3_1         ... done
  Stopping voting_mixnet_auth2_1        ... done
  Stopping voting_mixnet_mix2_1         ... done
  Stopping voting_mixnet_gateway1_1     ... done
  Stopping voting_mixnet_auth1_1        ... done
  Stopping voting_mixnet_mix1_1         ... done
  Removing voting_mixnet_auth3_1        ... done
  Removing voting_mixnet_servicenode1_1 ... done
  Removing voting_mixnet_metrics_1      ... done
  Removing voting_mixnet_mix3_1         ... done
  ```

```
Removing voting_mixnet_auth2_1        ... done
Removing voting_mixnet_mix2_1         ... done
Removing voting_mixnet_gateway1_1     ... done
Removing voting_mixnet_auth1_1        ... done
Removing voting_mixnet_mix1_1         ... done
removed 'running.stamp'
rm -vf ./voting_mixnet/*.alpine
removed './voting_mixnet/echo_server.alpine'
removed './voting_mixnet/fetch.alpine'
removed './voting_mixnet/memspool.alpine'
removed './voting_mixnet/panda_server.alpine'
removed './voting_mixnet/pigeonhole.alpine'
removed './voting_mixnet/ping.alpine'
removed './voting_mixnet/reunion_katzenpost_server.alpine
removed './voting_mixnet/server.alpine'
removed './voting_mixnet/voting.alpine'
```

This command leaves in place the cryptographic keys, the state data, and the logs.

- **clean-local**

  To delete both compiled binaries and data, run the following command:

  ```
  ~/katzenpost/docker$ make clean-local
  ```

  ```
  [ -e voting_mixnet ] && cd voting_mixnet && DOCKER_HOST=u
  Removing voting_mixnet_mix2_1         ... done
  Removing voting_mixnet_auth1_1        ... done
  Removing voting_mixnet_auth2_1        ... done
  Removing voting_mixnet_gateway1_1     ... done
  Removing voting_mixnet_mix1_1         ... done
  Removing voting_mixnet_auth3_1        ... done
  Removing voting_mixnet_mix3_1         ... done
  Removing voting_mixnet_servicenode1_1 ... done
  Removing voting_mixnet_metrics_1      ... done
  removed 'running.stamp'
  rm -vf ./voting_mixnet/*.alpine
  removed './voting_mixnet/echo_server.alpine'
  removed './voting_mixnet/fetch.alpine'
  removed './voting_mixnet/memspool.alpine'
  removed './voting_mixnet/panda_server.alpine'
  removed './voting_mixnet/pigeonhole.alpine'
  removed './voting_mixnet/reunion_katzenpost_server.alpine
  removed './voting_mixnet/server.alpine'
  removed './voting_mixnet/voting.alpine'
  git clean -f -x voting_mixnet
  Removing voting_mixnet/
  git status .
  On branch main
  Your branch is up to date with 'origin/main'.
  ```

- **clean**

  To stop the the network and delete the binaries, the data, and the go_deps image, run the following command as superuser:

```
~/katzenpost/docker$ sudo make clean
```

- **clean-local-dryrun**

```
~/katzenpost/docker$ make clean-local-dryrun
                       git clean -n -x voting_mixnet
                       Would remove voting_mixnet/
```

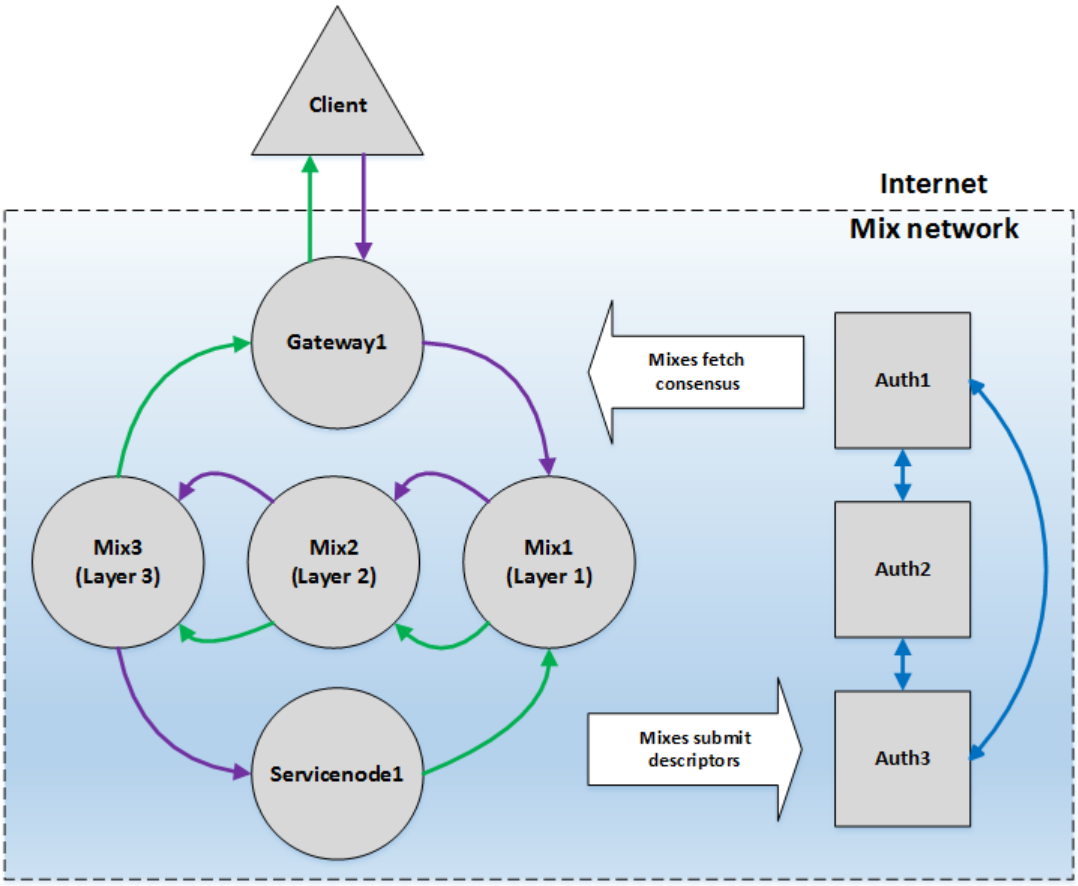
```
~/katzenpost/docker$ make clean
```

For a preview of the components that **clean-local** would remove, without actually deleting anything, running **clean-local-dryrun** generates output as follows:

Network components and topology

There needs to be an interpretation of this diagram. including the ways that the testnet differs from production network.

Figure 2.1. Test network topology



Discuss how to view these components, where their configuration files are, etc.

Table 2.2. Network hosts

Host type	Identifier	IP	Port	Panda
Directory authority	auth1	127.0.0.1	30001	
Directory authority	auth2	127.0.0.1	30002	
Directory authority	auth3	127.0.0.1	30003	
Gateway node	gateway1	127.0.0.1	30004	
Service node	servicenode1	127.0.0.1	30006	✓
Mix node	mix1	127.0.0.1	30008	
Mix node	mix2	127.0.0.1	30010	
Mix node	mix3	127.0.0.1	30012	