NAME

mixminion − Type III anonymity client

SYNOPSIS

mixminion {**benchmarks** | **clean-queue** | **decode** | **flush** | **generate-surb** | **help** | **import-server** | **inspect-queue** | **inspect-surbs** | **list-fragments** | **list-servers** | **ping** | **purge-fragments** | **queue** | **reassemble** | **send** | **shell** | **testvectors** | **unittests** | **update-servers** | **version**} [*command_options*][*command_args ...*]

Sending Messages:

mixminion send [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*][**-P** *path* | **--path=***path*] [**-i** *file* | **--input=***file*][**--queue** | **--noqueue**][**--subject=***str*] [**--from=***str*][**--references=***str*][**--in-reply-do=***str*] [**--deliver-fragments**]{**-t** *addr* | **--to=***addr* | **-R** *file* | **--reply-block=***file* | **--reply-block-fd=***n*}

mixminion queue [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*][**-P** *path* | **--path=***path*] [**-i** *file* | **--input=***file*][**--subject=***str*][**--from=***str*] [**--references=***str*][**--in-reply-do=***str*][**--deliver-fragments**]{**-t** *addr* | **--to=***addr* | *R file* | **--reply-block=***file* | **--reply-block-fd=***n*}

Receiving messages:

mixminion decode [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*][**-F** | **--force**][**-i** *file* | **--input=***file*] [**-o** *file* | **--output=***file*][**--passphrase-fd=***n*]

mixminion reassemble [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*][**-P** | **--purge**] [**-o** *file* | **--output=***file*][**-F** | **--force**]*msgid ...*

Queue Management:

mixminion flush [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*][*server ...*]

mixminion list-queue [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*]

mixminion clean-queue [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*][**-d** *days* | **--days=***days*][*server ...*]

mixminion list-fragments [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*]

mixminion purge-fragments [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**] [**-D** *{yes|no}* | **--download-directory=***{yes|no}*] [**-f** *file* | **--config=***file*]*msgid ...*

SURB Management:

mixminion generate-surb [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**]
[**-D** *{yes|no}* | **--download-directory=***{yes|no}*]
[**-f** *file* | **--config=***file*][**-P** *path* | **--path=***path*]
[**-lifetime=***days*][**-o** *file* | **--output=***file*][**-b** | **--binary**]
[**-n** *n* | **--count=***n*][**--identity=***name*][**--passphrase-fd=***n*]
[**-t** *addr* | **--to=***addr*]

mixminion inspect-surbs [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**]
[**-D** *{yes|no}* | **--download-directory=***{yes|no}*]
[**-f** *file* | **--config=***file*]*file ...*

Directories and Servers:

mixminion ping [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**]
[**-f** *file* | **--config=***file*]
[**-D** *{yes|no}* | **--download-directory=***{yes|no}*]*server ...*

mixminion list-servers [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**]
[**-f** *file* | **--config=***file*]
[**-D** *{yes|no}* | **--download-directory=***{yes|no}*]
[**-F** *feature_list* | **--feature=***feature_list*][**-J** | **--justify**]
[**-T** | **--with-time**][**-r** | **--recommended**]
[**-s** *str* | **--separator=***str*][**-c** | **--cascade**]
[**-C** | **--cascade-features**][**--no-collapse**]*server ...*

mixminion update-servers [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**]
[**-f** *file* | **--config=***file*]

mixminion import-server [**-h** | **--help**][**-v** | **--verbose**][**-Q** | **--quiet**]
[**-D** *{yes|no}* | **--download-directory=***{yes|no}*]
[**-f** *file* | **--config=***file*]*file ...*

Miscellaneous:

mixminion version
mixminion help
mixminion unittests
mixminion benchmarks
mixminion testvectors

NOTE

This is a testing alpha release. You will probably only want to use it if you are technically inclined, curious, and interested in helping the Mixminion development effort. Do NOT use this release if you require strong anonymity. It has known deficiencies, including some that make it possible for an adversary to trace your message through the system. See BUGS below.

DESCRIPTION

Mixminion is software suite that lets you send and receive very anonymous mail. The `mixminion`(1) client is a command-line interface to this suite.

If you want to send anonymous messages right now without reading all of this documentation, skip right to the EXAMPLES section below.

Overview

(This is a basic introduction. For more information, see REFERENCES below.)

When you send regular email, it is trivial for an eavesdropper to learn your recipients. Even if you use end-to-end encryption such as PGP or S/MIME, you are only stopping eavesdroppers from learning what you are saying: they can still tell whom you are saying it to.

Mixminion uses a *mix network* architecture to provide strong anonymity, and prevent eavesdroppers and other attackers from linking senders and recipients. Volunteers run servers (called "mixes") that receive messages, and decrypt them, re-order them, and re-transmit them toward their eventual destination. Every message passes through several mixes so that no single mix can link message senders with recipients.

When you send an anonymous message, **mixminion** breaks it into uniform-sized chunks (also called "packets"), pads the packets to a uniform size, and chooses a path through the mix network for each packet. The software encrypts every packet with the public keys for each server in its path, one by one. When it is time to transmit a packet, **mixminion** sends it to the first mix in the path. The first mix decrypts the packet, learns which mix will receive the packet, and relays it. Eventually, the packet arrives at a final (or "exit") mix, which sends it to your chosen recipient. Because no mix sees any more of the path besides the immediately adjacent mixes, they cannot link senders to recipients.

If you want to conceal the timing of your messages, **mixminion** can hold a group of packets in a *queue* until you are ready to transmit them. Packets will also be queued if the chosen servers are temporarily unavailable; you will need to retransmit these packets manually.

To choose good paths, **mixminion** must learn about all the available servers in the mix network. It does this by automatically downloading a *server directory* once every day it is used. You can examine this directory by hand, or with the **mixminion list-servers** command. You can also specify your own paths by hand.

Mixminion supports *Single-Use Reply Blocks* (or *SURBS*) to allow anonymous recipients. A SURB encodes a half-path to a recipient, so that each mix in the sequence can unwrap a single layer of the path, and encrypt the message for the recipient. When the message reaches the recipient, the recipient can decode the message and learn which SURB was used to send it; the sender does not know which recipient has received the anonymous message.

When you receive an anonymous message with **mixminion**, it may not be in plaintext. There are three kind of encoded messages:

1. Binary messages.
2. Anonymous reply messages send to SURBs addressed to you.
3. Single packets ("fragments") from large anonymous messages that span multiple packets.

The software handles each type differently. Binary messages are trivially decoded; anonymous replies are decrypted with your SURB key; and fragments are stored until enough fragments are available to reconstruct the entire message.

INVOKING MIXMINION

The **mixminion** interface wraps several commands, as listed below. To invoke a specific command, call **mixminion** *command_name*. Each command takes its own list of options.

Commands

Here is a brief description of each **mixminion** command:

send Break a message into packets, encode the packets, and send them into the mix network. By default, the message is read from standard input; type "control-D" when you are done ("control-Z" on Windows). To read the message from a file, use the **-i** *filename* ∞stored in the queue until you flush them.

mixminion send requires a destination; you can specify an address with the **-t** flag, or use one or more SURBs with the **-R** flag.

By default, the exit server will choose its own values for the outgoing message headers. You can override its selections with **--subject**, **--from**, **--refereneces**, and **--in-reply-to**.

queue Break a message into packets and encode them. Delivery is not attempted until you call **mixminion flush**. Using this command is equivalent to calling **mixminion send --queue**.

decode Extract the contents of an encoded message. If the message fits in a single packet, its contents will be printed out immediately; if the message is fragmented into multiple packets, the packets will be stored until the whole message has been received, at which point you can retrieve the original message with **mixminion reassemble**. If the packet is encrypted to a reply block, you will be prompted for a passphrase.

By default, **mixminion decode** reads from standard input and writes decoded messages to standard output; you can override input with the **-i** flag and output with the **-o** flag.

generate-surb[s]
Create one or more single-use reply blocks so that others can reply to your anonymous messages. By default, SURBs are written to standard output; you can override this with **-o** *filename*.

You may specify your address on the command line with the **-t** *addr* option, or in your configuration file (see mixminionrc(5)).

flush [*server ...*]
Try to flush packets from the queue. By default, **mixminion flush** tries to deliver every packet it can. If you only want to flush packets to a given set of mixes, you can list their names on the command line. You can also limit the number of packets flushed by using the **-n** *num* option to pick *num* packets at random from the queue.

Any packet that can't be delivered is left in the queue.

clean-queue [*server ...*]
Remove pending packets from the queue. By default, only very old packets are removed. You can use **-d** *days* option to remove packets older than a certain number of days. To remove packets for specific servers, specify their names on the command line.

reassemble *msgid*
Reassemble a fragmented message, and display its contents. By default, messages are written to standard output; you can override this with **-o** *filename*. Use **mixminion list-fragments** for a list of messages that can be reconstructed. Note that the packets reconstructed messages are not automatically deleted from disk: to do so, either use **--purge**, or the **mixminion purge-fragments** command.

list-fragments
Display a list of the messages currently being reconstructed, and the number of packets remaining until each it ready.

purge-fragments *msgid ...*
Remove fragments for a pending message. Use this command either after you have successfully reconstructed a message, or if you do not expect to receive any remaining fragments for the message.

inspect-queue
List the number of packets in the queue for each server, and the maximum age of each.

inspect-surbs *filename ...*
Examine a file containing a bunch of reply blocks. Lists how many have already been used, and when the others will expire.

ping *servername ...*
: Open a dummy connection to one or more named servers, to determine whether they are currently online.

  NOTE: Using this command has dangerous anonymity implications; see the output of **mixminion ping** for more information.

update-servers
: Download a fresh directory from the directory server, whether the current directory is out of date or not.

list-servers [*servername ...*]
: Display a list of current servers in the directory. By default, all servers are listed; you can narrow the list by using **-r** to see only servers that the directory recommends, or by listing specific servers on the command line.

  The default output lists one server per line, along with a summary of its capabilities and a note about whether the directory recommends it. You can change the contents of this display with **-F** *feature_list*, and the format with **-c**, **-C**, **-T**, **-s** *sep,* and **-J**.

import-server
: (Is anyone using this? I'm thinking of deprecating it to make my life simpler.)

help Print a command summary and exit.

testvectors Print a list of test vectors for Mixminion's cryptographic primitives. You may find this useful if you're trying to develop a compatible client.

unittests Run mixminion's internal self-tests.

version Print the version of the current software.

benchmark Run mixminion's internal timing tests.

shell Open a mini command-line interpreter for interactive operations. This is the default action on Windows when no command is provided.

Options

Here we describe the options supported by **mixminion**, along with a list of which commands support each one:

-b | **--binary**
: {generate-surb} Write surbs in a terser, binary format. By default, SURBs are printed with ASCII armor.

-c | **--cascade**
: {list-servers} List each server's name on a separate line from its validity dates and features. Especially useful in combination with **--no-collapse.**

-C | **--cascade-features**
: {list-servers} List each server feature on a separate line. Useful when you have specified a long list of features.

-d *days* | **--days=***days*
: {clean-queue} Only delete packets that are at least *days* days old. The default is 60.

-D *{yes|no}* | **--download-directory=***{yes|no}*
: {general} Override the directory download behavior. By default, mixminion downloads a fresh server directory if the current directory is older than midnight, GMT. If **-D** *yes* is specified, then

the software downloads a directory, even if it has a recent one. If **-D** *no* is specified, then the software uses the current directory, even if it is old.

--deliver-fragments
{send, queue} By default, mixminion sends (non-reply) fragmented messages to the last server in your path, and asks that server to reconstruct them before delivery. This way, users can receive large messages, even if they don't have the software to reconstruct it.

The **--deliver-fragments -option -overrides -this -behavior, -and -sends** the fragments directly to the recipient. This is useful if your message is large, and your recipient is running mixminion or a compatible client.

-f *file* | **--config=***file*
{general} Override the default location for your configuration file. See the MIXMINIONRC environment variable, and the mixminionrc(5) format.

-F | **--force**
{decode, reassemble} Mixminion compresses messages before sending them. Thus, a malicious person might try to mailbomb you by sending you highly compressed single packet that contained up to 28MB of compressed zeroes. By default, mixminion doesn't uncompress any file that has a compression ratio of higher than 20:1. The **-F** option overrides this behavior.

-F *feature_list* | **--feature=***feature_list*
{list-servers} Tells **mixminion list-servers** to display a group of features other than server capability and status. This list of servers is separated by commas. The default value is "caps,status". See "Server Features" below for a list of interesting features.

--from=*str*
{send, queue} Override the default value for the "From:" field in an outgoing message. Most exit servers let you replace the name portion of the message, but not the mailbox portion. For example, if you specify **--from=**Alice, the outgoing mail might have the line: 'From: "[ANON] Alice" <anonymous@example.com>'.

-h | **--help**
{general} Print usage information for a given command.

-i *file* | **--input=***file*
{send, queue, decode} Read input from a provided filename instead of standard input.

--identity=*name*
{generate-surb} Generate SURBs associated with a particular *identity*. Later, when you receive replies to a given SURB, you will be able to tell which identity the SURB was generated for.

Using this option can thwart so-called "identity blending attacks". For example, suppose that Alice is carrying on two pseudonymous conversations with Commissioner Bob, one as "Bruce Wayne" and one as "Batman". If Bob suspects that "Batman" and "Bruce Wayne" are really the same person, he can confirm this by using one of Bruce Wayne's SURBs to send a message addressed "Dear Batman". If "Batman" answers the message, then Bob's suspicion is confirmed. To prevent this, Alice should use separate identities for each of her pseudonyms, so that when "Bruce Wayne" receives the message, she can reply "I'm sorry, but you've send me one of Batman's messages."

--in-reply-to=*str*
{send, queue} Sets the "In-Reply-To:" header of an outgoing message.

-J | **--justify**
{list-servers} Align the columns of list-server's output nicely.

-lifetime=*days*
: {generate-surb} Generate SURBs to be used within the next *days* days. The default lifetime is *SURBLifetime* setting in your configuration file.

-n *n* | **--count=***n*
: {generate-surb} Generate exactly *n* SURBs. The default is 1.

: {flush-queue} Flush no more than *n* packets. By default, all packets are flushed.

--noqueue
: {send} If the packets can't be sent immediately, do not queue any failing packets. [**--no-collapse**] {list-servers} If a server has multiple keys valid over different ranges of time, display those ranges separately.

-o *file* | **--output=***file*
: {decode, generate-surb, reassemble} Direct output to *file* instead of standard output.

-P *path* | **--path=***path*
: {send, queue, generate-surb} Use *path* for the path of the packets/surbs being generated. The default is to use a randomly chosen path of about 4 mixes; you can override this in mixminionrc(5).

--P | **--purge**
: {reassemble} After reassembling the message, delete all its fragment packets from disk.

--passphrase-fd=*n*
: {decode, generate-surb} Instead of reading the passphrase from the terminal, take it from a given file descriptor. This is useful when embedding mixminion.

-Q | **--quiet**
: {general} Suppress all logging messages that don't correspond to real errors.

--queue
: {send} Do not actually send any packets; generate and queue them instead.

-r | **--recommended**
: {list-servers} Only list servers that the directory recommends. By default, all known servers are listed.

-R *file* | **--reply-block=***file*
: {send, queue} Send packets via the SURBs listed in *file*. *The file must contain one or more SURBs. Once a SURB* has been used, it will automatically be ignored in the future.

--references=*str*
: {send, queue} Set the "References:" header in the outgoing message.

--reply-block-fd=*n*
: {send, queue} Read reply blocks from the file descriptor *n* instead of from a file. This is useful when integrating mixminion in other applications.

--subject=*str*
: {send, queue} Set the "Subject:" header in the outgoing message.

-s *str* | **--separator=***str*
: {list-servers} When listing multiple features per line, separate them with the provided string. By default, a tab is used.

-t *addr* | **--to=***addr*
: {send, queue, generate-surb} Send messages or generate SURBs for the provided address. The address may be an email address (such as "somebody@example.com"), or a generalized address as described in "Specifying Destinations" below.

-T | **--with-time**
: {list-servers} Include the validity time ranges of each server when listing.

-v | **--verbose**
: {general} Display internal debugging messages when running the selected command.

Specifying Destinations

Although email delivery is Mixminion's current principal application, the client allows you to send messages to other destinations, such as newsgroups (not yet implemented), drop-boxes on given servers, or other protocols not yet specified. To do so, just provide a generalized address in place of an email address. The following address formats are supported:

email_address
: Sends an ordinary email message to a chosen email address.

smtp:*email_address*
: A verbose equivalent to sending email to *email_address*.

drop
: Send a dummy message into the message; the last mix in the path will discard it. When using this destination, no message body is needed.

mbox:*mailbox_name@server*
: Send a message to a named 'mailbox' configured at a specific mix. The chosen mix must support mbox delivery, and the name must be that of a valid mailbox.

0x*routing_type*
: For developers: manually send a message with a given 2-byte, hexadecimal routing-type field.

0x*routing_type*:*routing_data*
: For developers: manually send a message with a given 2-byte, hexadecimal routing-type field and a given routing_data field.

Note that not all servers support all exit types. Currently, mixminion only detects whether your chosen exit server supports smtp, exit, and drop delivery. You must use other means to detect whether other protocols are supported by a given server.

Specifying Paths

When you send a message or generate SURBs, mixminion ordinarily picks a series of mixes as your path. It tries to choose a sequence of about five mixes, such that every mix is recommended by the directory; no mix appears twice in a row; and the last mix is configured to deliver messages to your chosen destination.

You can override this default behavior on the command line using the **-P** *path* or **--path=***path* options, or in your configuration file. To specify a path for one of these options, provide a comma-separated list of:

~*number* Insert approximately *number* randomly chosen recommended servers.
∗*number* Insert exactly *number* randomly chosen recommended servers.
? Insert a single randomly chosen recommended servers.
server_name
: Insert a single server by name.

file_name Insert a server whose descriptor is stored in a separate file.

For example, **mixminion send -P A,B,C ...** sends a message through a path containing the server named A, the server named B, and the server named C. Running **mixminion send -P ~2,A,?,C ...** uses a path containing approximately two randomly chosen servers, the server named A, another randomly chosen server, and the server named C.

The software tries to never use the same server twice in a row, and never uses an un-recommended server unless such a server is specifically requested.

All forward paths must have at least two servers; all reply and SURB paths must have at least one. In practice, longer paths are recommended.

Advanced: For a forward path, you can specify a swap point by hand by replacing a single comma with a colon. If you don't know what a swap point is, you probably don't want to do this.

Server Features

When you use the **--F** option for **mixminion list-servers**, you can specify a list of server features. These features take the format of *section*:*entry*, where *section* is a section in a server descriptor, and *entry* is a key within that section. (For more information on server descriptors, see "http://mixminion.net/dir-spec.txt".) If the key is unique within all sections, then you may omit the *section*: portion of the feature.

A few useful features include:

caps The server's "capabilities", such as 'smtp', 'mbox', and 'frag'. This pseudo-feature is generated on the fly, and is not part of the server's descriptors.

status The string "(ok)" if the server is recommended, and "(not recommended)" if it is not.

contact An email address to contact the server's administrator.

contact-fingerprint
A PGP fingerprint for the server's administrator, if available.

server:comments A description of the server's status and policies.

software The Type III remailer software (and version) used by the server.

secure-configuration / why-insecure
Secure-configuration is true if the server is running in a believed-to-be-secure operating mode. If not, why-insecure explains what is insecure about the server's configuration. (Right now, all servers are believed-to-be-insecure, since the software is in alpha.)

IP The server's IP address, if known.

hostname The server's hostname.

delivery/mbox:maximum-size
The largest message in KB (before compression) that the server is willing to deliver via mbox.

delivery/smtp:maximum-size
The largest message in KB (before compression) that the server is willing to deliver via email.

delivery/smtp:allow-from
This value is true if the server supports client-selected from addresses.

maximum-fragments
The largest message size (in packets) that the server is willing to reassemble.

ENVIRONMENT

Mixminion recognizes the following environment variables:

`MIXMINIONRC` The default configuration for your `mixminionrc`(5) configuration file. Defaults to `$HOME/.mixminionrc` on Unix and Mac OS X, and *Documents and Settings\mixminionrc* on Windows. You can also override this default with the **`-f`** flag. If you try to start mixminion without a configuration, one is created.

`http_proxy` If you use a proxy to access the web, you should set this variable so that mixminion can use HTTP to download its directory.

`MM_NO_FILE_PARANOIA`
If set, don't check file permissions on private files.

FILES

Mixminion uses files as described below. Note: some of these filenames are given relative to a directory called "$BASE". This defaults to `$HOME/.mixminion/`, but you can override this with the *UserDir* setting in your `mixminionrc`(5).

`$HOME/.mixminionrc` Configuration options for **`mixminion`**, as documented in `mixminionrc`(5).

`$BASE/cache` A cached representation of the most recently downloaded directory, to speed server startup.

`$BASE/dir.gz` The most recently downloaded directory, compressed with gzip.

`$BASE/keys/` Secret keys used to decode messages sent to your SURBs. These keys are encrypted with your passphrase.

`$BASE/queue/msg_*` An outgoing packet in the queue, awaiting delivery.

`$BASE/queue/meta_*` Metadata for a single queued packet. (This includes the address of the first hop for a given queued packet, and the day on which the packet was generated.)

`$BASE/queue/rmv*_*` A packet that has been successfully delivered, and is waiting to be overwritten and removed.

`$BASE/fragments/` A directory of fragments for messages waiting reassembly.

`$BASE/fragments_db` A database describing which fragmented messages have been reassembled and/or purged. Future fragments for these messages will be automatically discarded.

`$BASE/imported/` A directory of server descriptors imported with **`mixminion import-server`**

`$BASE/surbs/log` A database of digests for SURBs which have been used already, to prevent repeat use. Entries are removed from this database once the corresponding SURBs are expired.

Note: the only one of these files you should ordinarily be modifying is `.mixminionrc`.

EXAMPLES

Send a message to an email address:

```
mixminion send -t <email address> -i <filename>
```

Send a message to an email address, reading from standard input:

```
mixminion send -t <email address>
```

Send a message to an email address, with a chosen "From" and "Subject" line:

```
mixminion send -t <email address> --from=Alice --subject="A subject"
```

Send a message to an email address, using the servers frell, noisebox, and moria:

```
mixminion send -t <email address> -P frell,noisebox,moria
```

Send a message to an email address, using about 5 randomly chosen servers, ending at moria:

```
mixminion send -t <email address> -P '~5,moria'
```

Send a message to an email address, starting at tonga, taking exactly three random steps, and ending at iabervon:

```
mixminion send -t <email address> -P 'tonga,*3,iabervon'
```

Send a message to an email address, starting at a random server, followed by totoro1, followed by a random server, followed by frell:

```
mixminion send -t <email address> -P '?,totoro1,?,frell'
```

Send a large message to be reassembled by the recipient (by default, the last server reassembles):

```
mixminion send -t <email address> --deliver-fragments
```

Reload the server directory immediately:

```
mixminion update-servers
```

Send a message to an email address, without reloading the directory:

```
mixminion send -D no -t <email address>
```

Send a dummy packet:

```
mixminion send -t drop
```

Queue a message to an email address for later delivery:

```
mixminion queue -t <email address>
```

Queue a dummy packet for later delivery:

```
mixminion queue -t drop
```

Test whether totoro1 is running:

```
mixminion ping totoro1
```

List all packets waiting in the queue:

```
mixminion inspect-queue
```

Try to flush all the packets in the queue:

```
mixminion flush
```

Try to flush no more than 25 packets from the queue:

```
mixminion flush -n 16
```

Remove packets that have been waiting in the queue for more than 7 days:

```
mixminion clean-queue -d 7
```

Decode a message you have just received, reading from one file and sending the output to another:

```
mixminion decode -i <input-file> -o <output-file>.
```

Decode a message you have just received, writing from standard input and writing to standard output:

```
mixminion decode -i -
```

Generate a single SURB, writing it to standard ouput:

```
mixminion generate-surb -t <your-address>
```

Generate 10 SURBs, writing them to a file:

```
mixminion generate-surb -t <your-address> -o surbs.txt
```

Generate a single SURB, using three randomly chosen servers then ending at frell:

```
mixminion generate-surb -t <your-address> -P '*3,frell'
```

Generate a SURB that will only be valid for 2 days:

```
mixminion generate-surb -t <your-address> --lifetime=2
```

Send a reply message to a user for whom we have a supply of SURBs stored in a file:

```
mixminion send -R surbs.txt
```

Examine the SURBs stored in a file:

```
mixminion inspect-surbs surbs.txt
```

List the servers currently running:

```
mixminion list-servers
```

List all the servers that are running Mixminion 0.0.7:

```
mixminion list-servers -F software | grep 'Mixminion 0.0.7'
```

REFERENCES

Mixminion's design is described and justified in more detail in the design paper:

- George Danezis, Roger Dingledine, and Nick Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol", *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003, http://mixminion.net/minion-design.pdf.

A byte-level specification of the Type III ("mixminion") remailer protocol, is available in the protocol specifications:

- George Danezis, Roger Dingledine, and Nick Mathewson, *MIX3:1 Type III (Mixminion) Mix Protocol Specification*, http://mixminion.net/minion-spec.txt.
- George Danezis, Roger Dingledine, and Nick Mathewson, *MIX3:2 Type-III Remailers: End-to-end Encoding and Delivery*, http://mixminion.net/E2E-spec.txt.
- George Danezis, Roger Dingledine, and Nick Mathewson, *MIX3:3 Type III (Mixminion) Mix Directory Specification*, http://mixminion.net/dir-spec.txt.

The path syntax is documented in

- Nick Mathewson and Peter Palfrader, *MIX3:path Type III (Mixminion) Path Generation*, http://mixminion.net/path-spec.txt.

There are also tenative specifications for an API and a nymserver, but this software implements neither. For more information, see the Mixminion website at http://mixminion.net/

SEE ALSO

`mixminionrc`(5), `mixminiond`(8)

AUTHORS

George Danezis, Roger Dingledine, and Nick Mathewson did the first Mixminion protocol design, which was later adopted for the Type III remailer protocol. Nick Mathewson wrote most of the software, with help from (many!) others.

Nick Mathewson ⟨nickm@freehaven.net⟩ wrote the first version of the documentation (a README file); George Danezis ⟨George.Danezis@cl.cam.ac.uk⟩ did the first version of the manpage; and then Nick revised it again. Any inaccuracies or omissions are probably Nick's fault.

ACKNOWLEDGMENTS

The Mixminion software is by Nick Mathewson, with contributions by Roger Dingledine, Brian Fordham, Lucky Green, Peter Palfrader, Robyn Wagner, Brian Warner, and Bryce "Zooko" Wilcox-O'Hearn.

BUGS

Future releases will probably break backward compatibility with this release at least once or twice.

Windows support is not as well-tested as it should be.

You shouldn't trust Mixminion with your anonymity yet, for the following reasons:

1. The code is under development. There may be unknown bugs that could compromise your anonymity. (We do not know know of any such bugs.
2. In order to test the code, many servers are running in configurations that could harm your anonymity. For example, some servers are configured to log verbosely. Others are configured to use the "timed-pool" mixing algorithm rather than the more robust "timed dynamic-pool" mixing algorithm. While these configurations help us debug Mixminion, they also make it easier for an eavesdropper or a compromised server to trace your messages. The final Mixminion release will not support these configurations.
3. Some features that are necessary for high security, robustness, anonymity are not yet implemented. These include:
   - Distributed directories. (The current centralized directory is a single point of failure.)
   - Automatic generation of dummy messages.
   - Built-in network reliability testing ("pinging").

Reporting bugs

To report bugs, please use the Bugzilla pages at http://bugs.noreply.org/. For other correspondence, please email <nickm@freehaven.net>. For help in debugging, please try to send a copy of:

- What command you were running.
- The complete error you got, including stack trace (if any).

If your error occurred on a running server, please make a copy of your log--it might be helpful.