nymphemeral Documentation

Release 1.3.5

Felipe Dau and David R. Andersen

September 10, 2015

CONTENTS

CHAPTER

ONE

WHAT IS NYMPHEMERAL?

1.1 Overview

nymphemeral is a tool made for users searching for secure and anonymous communication on the internet.

It is a GUI client that relies on a pseudonymous remailer that communicates to its users by posting messages to a shared mailbox, a Zax-type nym server. Both the server and the client apply an ephemeral encryption layer on their messages based on the Axolotl Ratchet protocol, providing forward and future secrecy to the conversation.

1.1.1 Features

- Manages pseudonymous actions: creation, configuration and deletion, as well as message dispatch and retrieval
- Communicates with the new nymserv, a *Zax-type* nym server with forward secrecy
- Uses python-gnupg and pyaxo for encryption
- Uses aampy to retrieve messages from alt.anonymous.messages
- Sends messages through Mixmaster, *sendmail*, or outputs the resulting ciphertexts to be sent manually
- Supports End-to-End Encryption

1.1.2 Current Release

The current version of nymphemeral is 1.3.5, a beta, released 2015-09-09.

1.1.3 Limitations

Regular Zax-type

nymphemeral does not support the regular Zax-type nym server. It only supports the new nymserv, adding or expecting an ephemeral encryption layer in its messages.

Mixmaster

Although it is supported (and the use is encouraged), nymphemeral is not a *Mixmaster* GUI. It does enable the users to send their messages to the nym server automatically via *Mixmaster*, but it cannot be used to send regular email. nymphemeral is a **nym client** and the only way to exchange messages is to send every message to the nym server, to be processed and then remailed to the recipient. *Mixmaster* is just one of the output methods.

Important: **nymphemeral 1.3.3** was updated to use **pyaxo 0.4** that follows the latest (Oct 1, 2014) version of the protocol, which changed the order of the ratcheting. For that reason, old conversations (created with **nymphemeral < 1.3.3**) might not work properly after the update. We suggest that users update nymphemeral and restart their conversations by changing their nyms' **ephemeral keys**. The *Configuring the Nym* section explains how that can be done.

CHAPTER

TWO

INSTALLATION (ON A DEBIAN WHEEZY/UBUNTU TRUSTY SYSTEM)

2.1 Main Dependencies

If you use pip, install nymphemeral with:

```
sudo pip install nymphemeral
```

The dependencies will be automatically downloaded and installed. You can go to *Other Dependencies*.

If you do not use *pip*, first make sure that you have the following:

```
sudo apt-get install python-dev python-tk
```

nymphemeral also uses pyaxo, python-dateutil and python-gnupg, and the easiest way to install those is using setuptools. After making sure you have *setuptools*, from nymphemeral's source folder, install with:

```
sudo python setup.py install
```

The dependencies will be installed automatically.

If you do not use *setuptools* as well, you will have to install each dependency and sub-dependencies manually.

2.1.1 Other Dependencies

nymphemeral will be ready for use after installation via either of the two methods described in *Main Dependencies*. However, you should follow the instructions from *Connections*, install *Mixmaster* and have a *News Server* running to be able to use all of its features.

2.2 Connections

We recommend using stunnel, Tor and socat along with nymphemeral when downloading and sending messages. If you are a Whonix user, you should go to connections-whonix.

2.2.1 Stunnel

stunnel adds *TLS* to your connections. You can install it with:

```
sudo apt-get install stunnel4
```

To configure *stunnel*, you can use the `.conf` file we provide with nymphemeral. Copy that file to the directory where *stunnel* looks for config files (which is usually `/etc/stunnel`):

```
sudo curl https://raw.githubusercontent.com/felipedau/nymphemeral/master/connections/stunnel.conf -o
```

Open `/etc/default/stunnel4` and enable *stunnel* automatic startup by switching `ENABLE` to `1`:

```
# Change to one to enable stunnel automatic startup
ENABLED=1
```

And start it with:

```
sudo service stunnel4 start
```

You should get the following message:

```
Starting SSL tunnels: [Started: /etc/stunnel/stunnel.conf] stunnel.
```

Tunelling

From the last sections of the `.conf` file:

```
[nntps-client]
client = yes
accept = 127.0.0.1:119
connect = 127.0.0.1:10063

[ssmtp-client]
protocol = smtp
client = yes
accept = 127.0.0.1:2525
connect = 127.0.0.1:2526
```

Note that:

- The NNTP client is used to download messages. Whenever it accesses port `119`, *stunnel* will connect it to port `10063`, adding *TLS*.
- The SMTP client is used to send messages. Whenever it accesses port `2525`, *stunnel* will connect it to port `2526`, adding *TLS*.

2.2.2 Tor

Tor is a low-latency communication system used to hide your traffic. If you wish to have the latest stable version you should use this option. If that is not the case, simply install it with:

```
sudo apt-get install tor
```

2.2.3 Socat

socat manages the last part of the process, which will make the connections from your machine to the servers via *Tor*. You can install it with:

```
sudo apt-get install socat
```

A script should be used to make the connection itself. Copy both *socat* scripts we provide with nymphemeral:

```
curl https://raw.githubusercontent.com/felipedau/nymphemeral/master/connections/socnews.sh -o ~/socne
curl https://raw.githubusercontent.com/felipedau/nymphemeral/master/connections/socsmtp.sh -o ~/socsm
```

And enable them to be executed:

```
chmod +x ~/socnews.sh ~/socsmtp.sh
```

News Server

From the `socnews.sh` file:

```
socat TCP-Listen:10063,bind=localhost,fork SOCKS4A:localhost:news.mixmin.net:563,socksport=9050 > /de
```

Note that *socat* accepts connections through port `10063` (the one that *stunnel* connects to) and then connects to the news server at *mixmin.net* via *Tor* through port `9050`.

Run it with:

```
~/socnews.sh
```

SMTP Server

From the `socsmtp.sh` file:

```
socat TCP-Listen:2526,bind=localhost,fork SOCKS4A:localhost:lnwxejysejqjlm3l.onion:2525,socksport=905
```

Note that *socat* accepts connections through port `2526` (the one that *stunnel* connects to) and then connects to the Jeremy Bentham Remailer SMTP server at anemonee.mooo.com via *Tor* through port `9050`.

Run it with:

```
~/socsmtp.sh
```

You could also use other SMTP servers, such as these ones:

```
mail.mixmin.net
mail.allpingers.net
```

Note: You can use whatever NNTP/SMTP servers you would like. We chose to use those for convenience, but you are totally free to configure other ones or setup your own.

Important: You do not need to start *stunnel* or *Tor* again, but the scripts have to be executed every time the system starts up or whenever you wish to use nymphemeral.

2.3 Mixmaster

This section describes how to compile the new large-key version of *Mixmaster* on a *Debian Wheezy* system. If you already have **Mixmaster 3** installed and configured you can go to *Pre-installed Mixmaster*. If you are a Whonix user, you should go to mixmaster-whonix.

Most of the content of this section was taken from this post by the Jeremy Bentham Remailer Admin. The instructions should be helpful for building *Mixmaster* on other flavors of linux as well. See *Ubuntu Loader Changes* for a change if using *Ubuntu*.

2.3.1 Preliminaries

First, you need to install the packages required by *Mixmaster* and *OpenSSL*:

```
sudo apt-get install build-essential libpcre3-dev wget \
zlib1g-dev libncurses5-dev curl perl bc dc bison libbison-dev
```

2.3.2 Build OpenSSL

Then you need to compile a version of *OpenSSL* that contains the *IDEA* cipher. Grab the most recent version (make sure it is version 1.0.1g or later!) from the OpenSSL download page.

Extract the tarball (substituting your version for 1.0.1g):

```
tar xvf openssl-1.0.1g.tar.gz
```

Build the distribution:

```
cd openssl-1.0.1g
./config
make
make test
sudo make install
```

Note that this installs *OpenSSL* into `/usr/local/ssl`. Symlink the new *OpenSSL* installation into your normal lib and include directories so that the *Mixmaster* install script can find them. Note that the `sudo mv` instructions below will only work if you have previous copies of the files installed. If you get an error along the lines of `mv: cannot stat libssl.a` or similar, just ignore it - you did not have a file there to move:

```
cd /usr/lib
sudo mv libssl.a libssl.a.old
sudo ln -s /usr/local/ssl/lib/libssl.a libssl.a
sudo mv libcrypto.a libcrypto.a.old
sudo ln -s /usr/local/ssl/lib/libcrypto.a libcrypto.a
cd /usr/include
sudo mv openssl openssl.old
sudo ln -s /usr/local/ssl/include/openssl openssl
```

2.3.3 Build Mixmaster

Download Mixmaster 3.0.3.

Be sure to verify the SHA256 hash of the downloaded file. You can do this by executing the command:

```
sha256sum mixmaster-3.0.3b.tar.gz
```

The output should match the following hex number:

```
4cd6121e49cddba9b0771d453fa7b6cf824bee920af36206d1414388a47708de
```

Extract the *Mixmaster* tarball:

```
tar xvf mixmaster-3.0.3b.tar.gz
```

Run the `Install` script:

```
cd mixmaster-3.0.3b
./Install
```

Answer the questions posed by the script:

- You can just press enter when it prompts for the installation directory. It will be installed at `~/Mix`
- Do not worry about the *OpenSSL* version questions - 1.0.1g+ is so new the script does not know about it - select the default **YES**
- Your new version of *OpenSSL* **does** have AES encryption, so answer **YES** to that question as well
- As we are going to only use *Mixmaster* as a client (with nymphemeral), answer **NO** to the question about running a remailer

Mixmaster should be installed successfully.

Ubuntu Loader Changes

If you are using *Ubuntu* and see the following compile error:

```
gcc mix.o rem.o rem1.o rem2.o chain.o chain1.o chain2.o nym.o pgp.o pgpdb.o pgpdata.o pgpget.o pgpcre
/usr/bin/ld: /usr/local/ssl/lib/libcrypto.a(dso_dlfcn.o): undefined reference to symbol 'dlclose@@GLI
/lib/x86_64-linux-gnu/libdl.so.2: error adding symbols: DSO missing from command line
collect2: error: ld returned 1 exit status
make: *** [mixmaster] Error 1
Error: The compilation failed. Please consult the documentation (section `Installation problems').
```

you should make the following changes to the `Install` script, due to modifications *Ubuntu* has made to the loader.

On line `402` of the `Install` script, change:

```
LDFLAGS=
```

to:

```
LDFLAGS="-ldl"
```

2.3.4 Getting New Remailer Stats

Before you can use *Mixmaster*, you need to update the stats. We are going to use the pinger from the Jeremy Bentham Remailer, but the process should be similar to other pingers you wish to use.

An easy way to do this **securely** is with *curl*. First, create a file called `update.sh` in your `~/Mix` directory, with the following contents:

```
#!/bin/bash
export SSL_CERT_DIR=$HOME/Mix/certs
rm pubring.asc pubring.mix mlist.txt rlist.txt
curl --cacert ./certs/anemone.pem https://anemone.mooo.com/stats/mlist.txt -o mlist.txt
curl --cacert ./certs/anemone.pem https://anemone.mooo.com/stats/rlist.txt -o rlist.txt
curl --cacert ./certs/anemone.pem https://anemone.mooo.com/stats/pubring.mix -o pubring.mix
curl --cacert ./certs/anemone.pem https://anemone.mooo.com/stats/pgp-all.asc -o pubring.asc
```

Change the script to executable mode:

```
chmod +x update.sh
```

Next, create the `~/Mix/certs` directory and add *anemone.mooo.com's* certificate:

```
mkdir ~/Mix/certs
cd ~/Mix/certs
wget http://anemone.mooo.com/anemone.pem
```

Now that you have downloaded the certificate file, you can securely update your remailer stats by simply:

```
cd ~/Mix
./update.sh
```

You should update the remailer stats *at least once a day* when using *Mixmaster*.

2.3.5 Config File

Mixmaster just needs to be configured through the `~/Mix/mix.cfg` file. A very simple config file could be written as follows:

```
CHAIN *,*,*,*,*
SMTPRELAY localhost
SMTPPORT 2525
HELONAME anonymous.invalid
REMAILERADDR anonymous@anonymous.invalid
```

Chain

The `CHAIN` is the path that your messages will take before being delivered. In the configuration above, the messages are going to pass by five mixes, and finally get to the actual target. You can use any sequence and number of mixes in the chain, passing their names or simply `*` (which means that it could be any mix), separated by commas.

Note: Adding more mixes to the chain will probably increase the latency to deliver your messages. That is actually not a bad thing, but you should decide how long you are willing to wait to exchange messages.

SMTP Server

If you followed *Connections*, you remember that we will use port `2525` to reach an SMTP server. Using the options `SMTPRELAY` and `SMTPPORT` will tell *Mixmaster* to use that specific connection. Finally, as part of the protocol you need to provide a `HELONAME` and a `REMAILERADDR`. As we want to be anonymous, we provide an invalid address.

Note: nymphemeral should be ready to tunnel via Tor messages sent using Mixmaster!

2.3.6 Pre-installed Mixmaster

Although we encorage the use of the *Mixmaster* version installed with this section, improved with **4096-bit RSA** (and other features), you are allowed to use any derivative of **Mixmaster 3**. As long as you use that version and nymphemeral is able to find both paths to the binary and config file, you are fine. Configuring these paths is explained later on *[mixmaster]*.

2.4 News Server

Zax-type nym servers deliver messages to their nyms by posting them on a news group. aampy is the tool underneath nymphemeral that downloads those messages via a news server. The default news server configured in `nymphemeral.cfg` is set to `localhost`, port `119`. This default is useful if you use stunnel to encrypt the connection between `localhost:119` and your actual news server, exactly what was done on *Connections*. If you followed that section, you do not need to configure anything.

Note: nymphemeral should be ready to tunnel your news feed via Tor!

If you want to connect directly to the news server, you should edit `nymphemeral.cfg` and change the address and port of the news server appropriately. Unfortunately, the **python 2.7 nntplib** module does not support connections over SSL/TLS.

CHAPTER

THREE

USAGE

3.1 Files Structure

After installing the client, run nymphemeral at the command line and it will create the following files and directories inside the base directory (`~/.config/nymphemeral`):

- `*.gpg`: *GPG* keyring files
- `nymphemeral.cfg`: Config file that stores preferences and paths used by the client. This file is not encrypted and does not have sensitive data
- hSub files that store the hSub passphrases of the nyms
  - `encrypted_hsubs.txt`: File encrypted to every nym (with asymmetric encryption) by one that has access to it
  - `hsubs.txt`: File used to store (temporarily) the hSub passphrases of new nyms that still do not have access to the encrypted one
- `db`: Database directory that stores the conversation states of all the nyms. These databases are protected with symmetric encryption (using the passphrases the user provided when creating each nym)
- `messages`: Directory that stores the read and unread messages
  - `unread`: Directory that stores the messages downloaded from the news group, that are already encrypted with ephemeral encryption from the server
  - `read`: Directory that stores the messages the user chose to save, that are encrypted with asymmetric encryption, where the nym encrypted to itself

3.1.1 Configuring nymphemeral

You can modify `nymphemeral.cfg` per your liking. We will not describe the whole file, but only the options relevant to the user that belong to the following sections:

[gpg]

Although this option can be modified through the GUI, you can toggle `use_agent` between `True`/`False` to use the GPG Agent when signing/decrypting messages. (Default: `True`)

[main]

The value of `logger_level` can be modified to control what nymphemeral logs on the console. These values are the same ones used by Python's logging module. You can choose from:

debug
info
warning
error
critical

Example: `debug` is the most sensitive level. When it is set, every message will be logged from `debug` to `critical`. (Default: `warning`)

[mixmaster]

This section defines the paths nymphemeral searches for *Mixmaster's* binary and config file. Values defined on this section will be checked before the default paths *Mixmaster* usually uses for both compiled (as instructed on *Mixmaster*) and installed (with the package manager) versions:

Option	Compiled	Installed
binary	*~/Mix/mixmaster*	*mixmaster*
cfg	*~/Mix/mix.cfg*	*~/.Mix/mix.cfg*

If your *Mixmaster* installation is different from these values, you must change the `binary` and `cfg` options accordingly. nymphemeral calls `--version` on the binary and checks for the existence of the config file. Only after checking that *Mixmaster* is some derivative of **Mixmaster 3** and the config file is found, it assumes it is installed and working. Finally it searches for the *mix chain* to be displayed on the GUI, but will not prevent *Mixmaster* to be used if it is not found.

[newsgroup]

If you already have a news server running, replace `group`, `server` and `port` with its information. Otherwise, visit *News Server* to find out how to create one using *socat* and *stunnel*.

Important: Changes made to `nymphemeral.cfg` will only take effect by restarting the client.

3.2 GPG Keyring

nymphemeral has its own GPG keyring within its base directory and it does not access information from the user keyring. Therefore, if you are going to use End-to-End Encryption, you have to manually add the respective key to the keyring.

More information regarding End-to-End Encryption can be found in the *Sending Messages* and *Decrypting Messages* sections.

3.2.1 Adding Key

Considering you will encrypt a message to a user whose public key is in the `pkey.asc` file in the home directory. You can add it to nymphemeral's keyring with:

```
gpg --homedir ~/.config/nymphemeral --import ~/pkey.asc
```

Now you can type its UID or fingerprint when encrypting the message.

Similarly, you can also add private keys to the keyring if you expect to receive messages encrypted to a specific key you have. Either the GPG Agent or nymphemeral will automatically prompt you for a passphrase and decrypt the message.

3.3 Starting Session

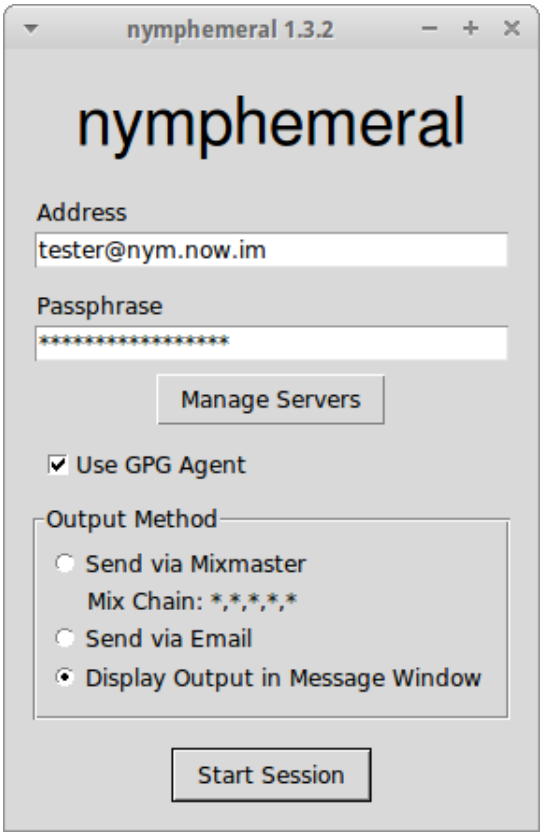

Fig. 3.1: Login Window

When you run the client, a login window will be displayed. Fill in the `Address` and `Passhphrase` fields, choose the output method that you would like to use and click `Start Session`.

3.3.1 GPG Agent

If it is enabled, the GPG Agent's dialogs will be displayed when you need to sign/decrypt messages, prompting you for a passphrase. If you decide not to enable it, nymphemeral's own dialogs will be used.

3.3.2 Output method

When **Mixmaster** is installed and configured, clicking the `Send via Mixmaster` radio button on the login screen will route all messages to the nymserv through the Mixmaster network automatically.

If you have **sendmail** configured and running on your machine, you can also choose to send messages to the nymserv as regular email via the `Send via Email` radio button automatically.

If you would rather send messages manually, select the `Display Output in Message Window` radio button and then copy the encrypted message from the message window for transmission. If you choose this option it is your responsibility to send the encrypted message to the server. When this last method is being used, the client assumes that the message will get to the server. Therefore, when you finish the creation process, the nym information will be written to disk right away as well as it will be deleted when you confirm to delete the nym.

Important: Regardless the method that is being used, information about the message that has just been created is displayed in the first lines of the text box from the current tab.

3.3.3 Managing Servers

If the nymserver's public key is not found in the keyring, you will be prompted to add it. You can also add, modify or delete these public keys whenever you want by clicking on `Manage Servers` in the login window.

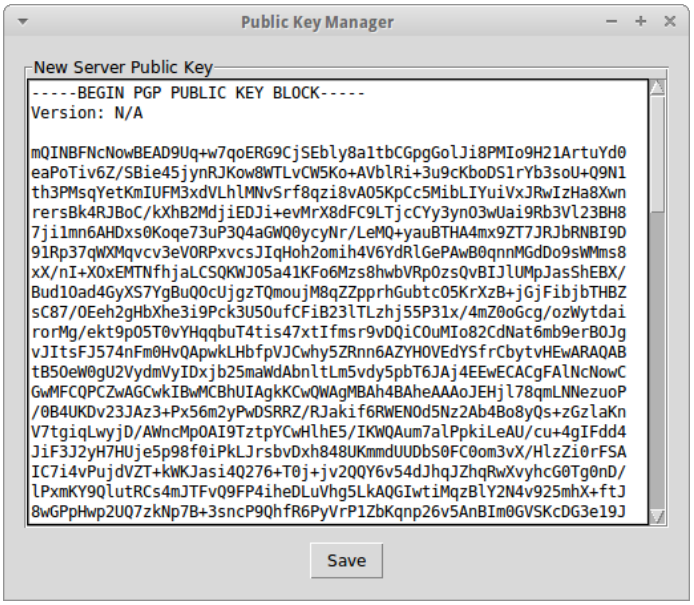

Fig. 3.2: Key Manager Window

3.4 Creating a Nym

Back to the login window, when you click `Start Session` and the nym server's public key is already in the keyring, then the client will search for the nym address that was given. If the nym is not found, you will be asked if you wish to create it and you will be directed to the `Create Nym` tab in the main window. To create a nym, you must provide the following information:

3.4.1 Ephemeral Key

The Axolotl Ratchet protocol derives a master key from the handshake to start a conversation. Since there is already a secure channel between you and the server (using its PGP key), the user can go ahead and send a master key to

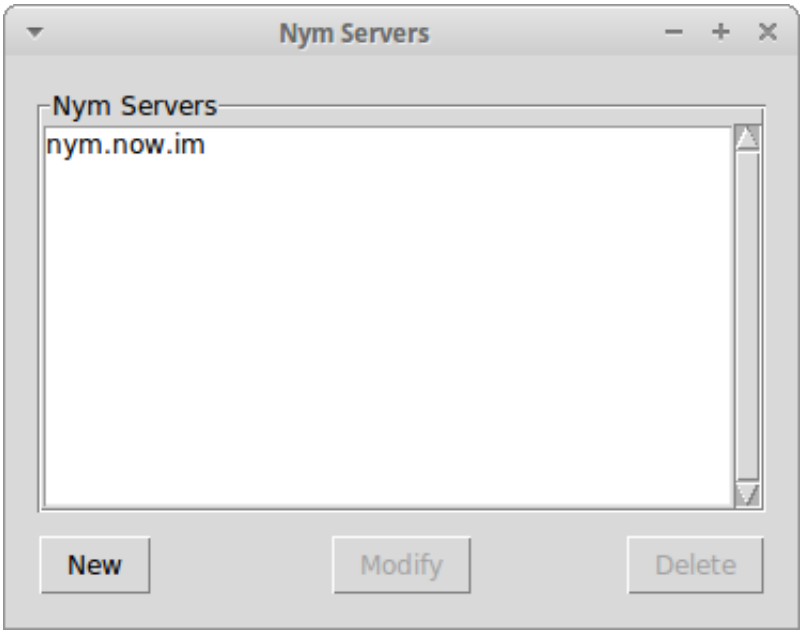

Fig. 3.3: Server Manager Window

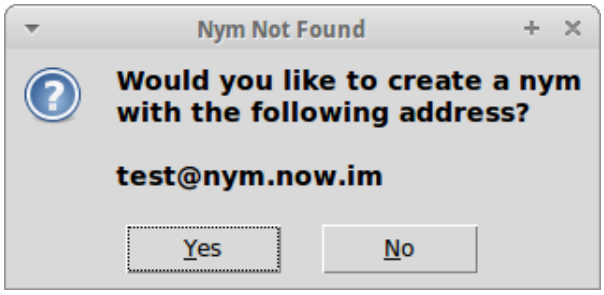

Fig. 3.4: Nym Not Found Dialog

skip that extra step. That key is the `Ephemeral Key` you need to provide. It can be any random string since it is ephemeral and will be used only on the first round of message exchange until the ratcheting starts.

3.4.2 hSub Passphrase

Using a **hashed subject (hSub)** is the easiest way of setting up subject identification for your nym to retrieve messages. An hSub is made of two parts, where the first is a random number and the second is the hash of that same random number and a passphrase. As the hashing is a one-way function, no one can identify the owner of the message. However, as you know your nym's hSub passphrase, you can hash it with the random number of every message, and if the result collides with the second part of the hSub, that message was sent to your nym. More on hSub by Zax.

Saving your hSubs allows nymphemeral to retrieve messages from all your nyms at once. You just need to know how the encryption of the hSub passphrases file works:

The first nym that you create will encrypt its hSub passphase and can only be decrypted by itself. The next nym to be created will save its hSub passphrase in plaintext and will not be able to access the encrypted file until you re-log in with the first nym. Then, it will encrypt both passphrases to both nyms and if you create a third nym, those two other nyms can encrypt the passphrases to the third one and so on.

Note: Although the `hSub Passphrase` is not required to use a nym, this client works better if you use one and we decided to make it a required field. If you feel that it should allow nyms without an hSub, let us know.

3.4.3 Pseudonymous

The `Pseudonymous` field is the name you are going to give to the nym.

3.4.4 Duration

You must provide the duration of the nym's key. Once the key expires, the nym expires as well. The `Duration` must be in the same format used by **GPG** (e.g. `1w`, `2m`, or `3y`).

3.4.5 Create Nym

Finally, click `Create Nym`. The text box will display the output message. Read it to see if the message was sent successfully. The nym will be created and the other tabs will be enabled.

3.5 Decrypting Messages

After the nym is created, go to the `Inbox` tab. There you can click `Retrieve Messages` to start **aampy** and if there are messages sent to you that are tied to your hSub key, they will be displayed in the `Messages` list box. Clicking on one of the messages will decrypt it and display it in the `Body` tab and its headers in the `Headers` tab. If you wish to keep the message, click `Save to Disk`. You can also click `Reply Message` and you will be directed to the `Send Message` tab.

Note: Clicking on undecrypted messages will delete them when the process is done. If it fails to decrypt then it is useless. If it succeeds, then it becomes useless due to the Axolotl ratchet protocol. You can go to pyaxo to read more about it.

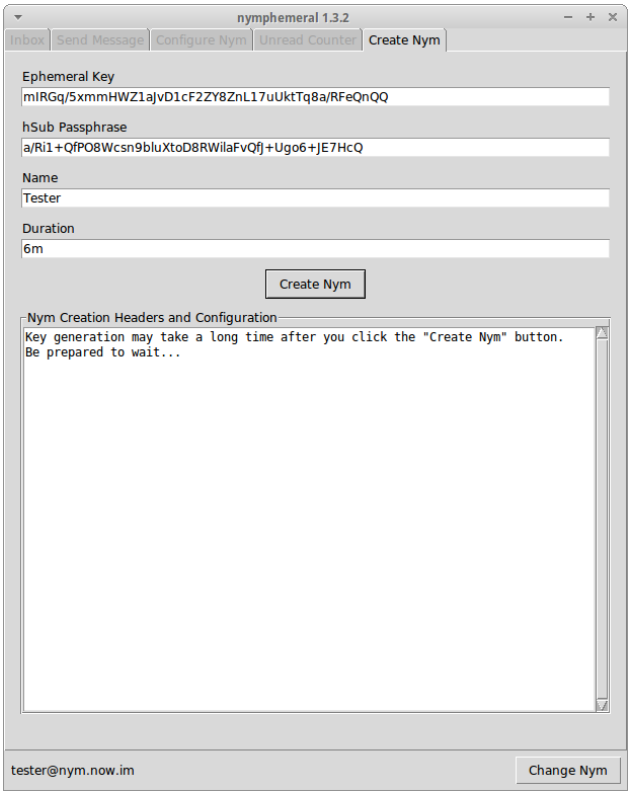

Fig. 3.5: Create Nym Tab

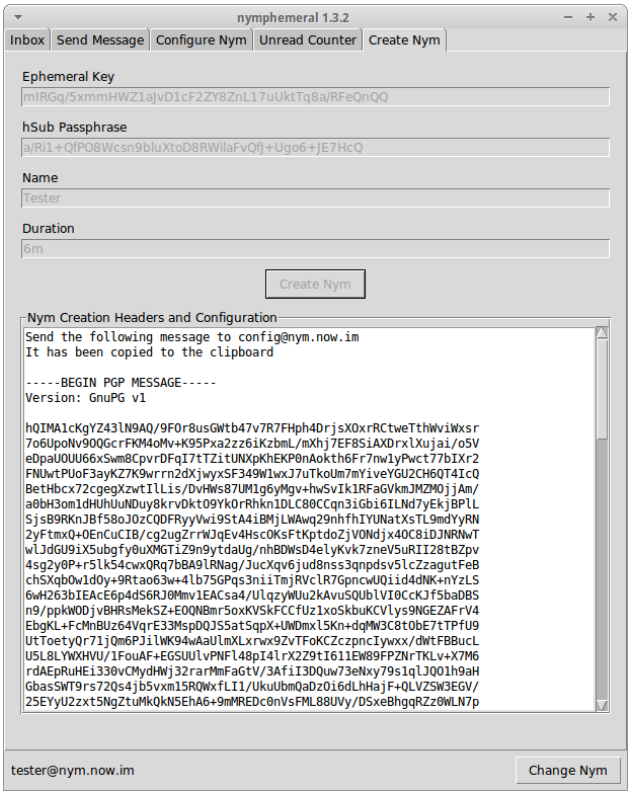

Fig. 3.6: Creation Message

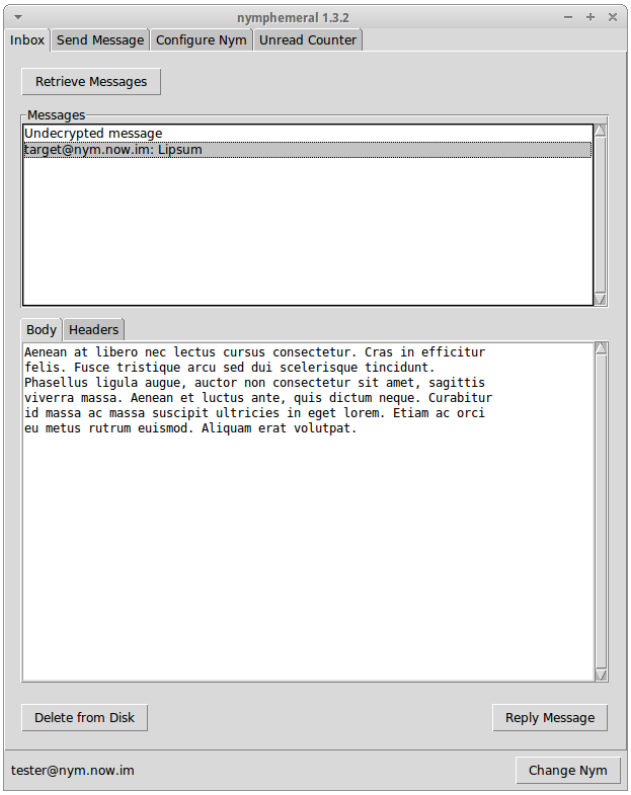

Fig. 3.7: Body Tab

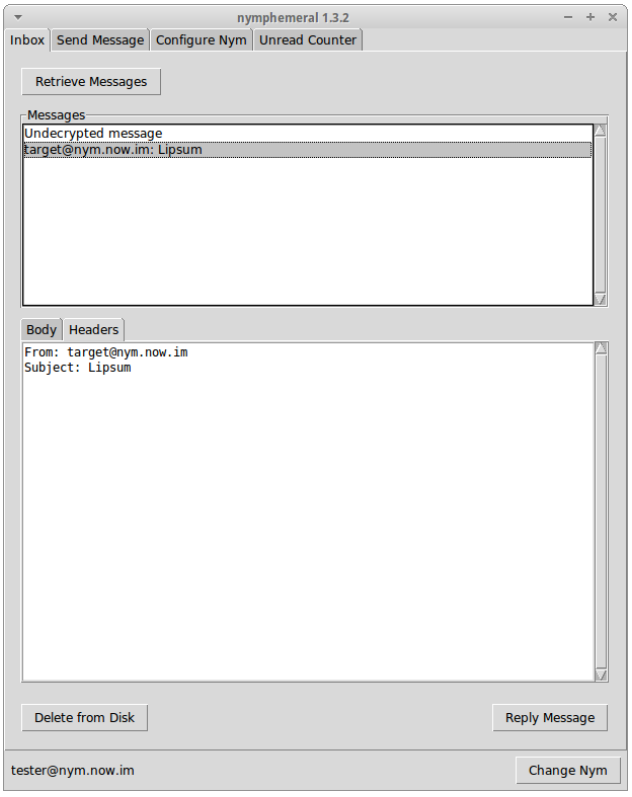

Fig. 3.8: Headers Tab

3.5.1 End-to-End Encryption

If someone has sent you an End-to-End Encrypted message, when you click on the message to decrypt it, after removing the encryption layers that were added by the nym server, either the GPG Agent or nymphemeral will prompt you for a passphrase if the key is found in the keyring:

Fig. 3.9: GPG Agent

If the key is not found or the authentication fails, the ciphertext will be displayed, allowing you to decrypt it manually.

Note: You should read the *GPG Keyring* section to add the keys involved in the End-to-End Encryption to nymphemeral's keyring.

3.6 Sending Messages

Sending a message is simple. Fill in the `Target Email Address`, `Subject` and `Message` fields and click `Send`.

3.6.1 Optional Headers

In the `Headers` text box, other headers can be added to the message in the format:

```
HeaderA: InformationA
HeaderB: InformationB
```

Example

I know a server that allows me to post messages to *Usenet*. I provide its email address in the `Target Email Address` and as I wish to post to *alt.privacy.anon-server*, I type the following header in the `Headers` text box:

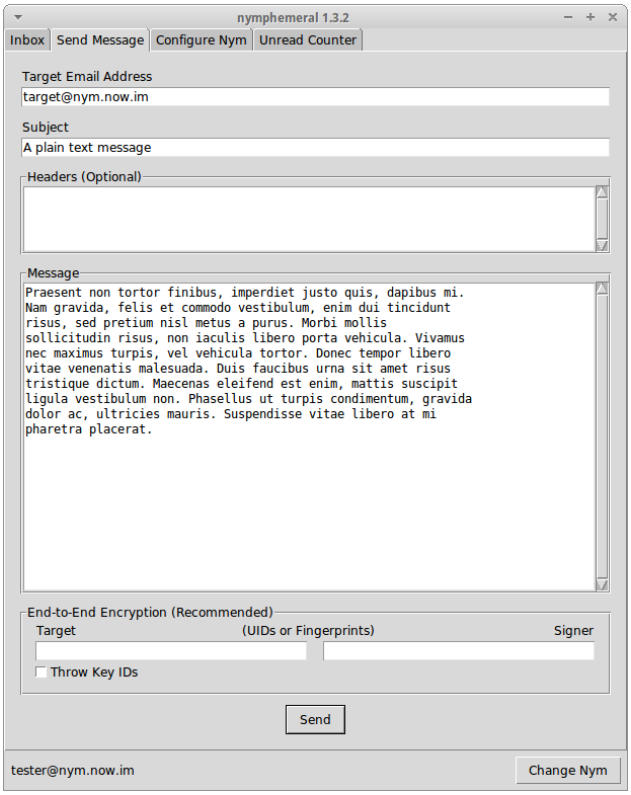

Fig. 3.10: Send Message Tab

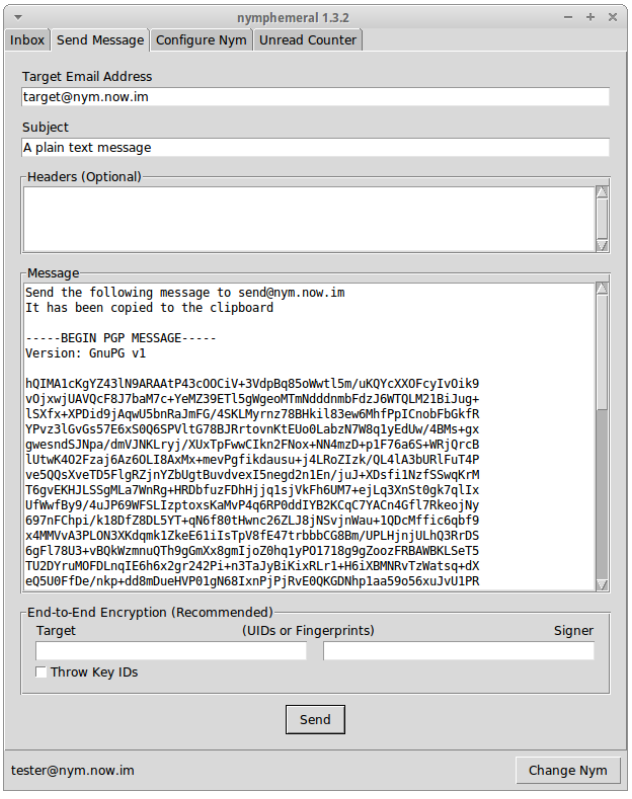

Fig. 3.11: Sent Message

```
Newsgroups: alt.privacy.anon-server
```

The server will process the message and the post should arrive in the news group.

3.6.2 End-to-End Encryption

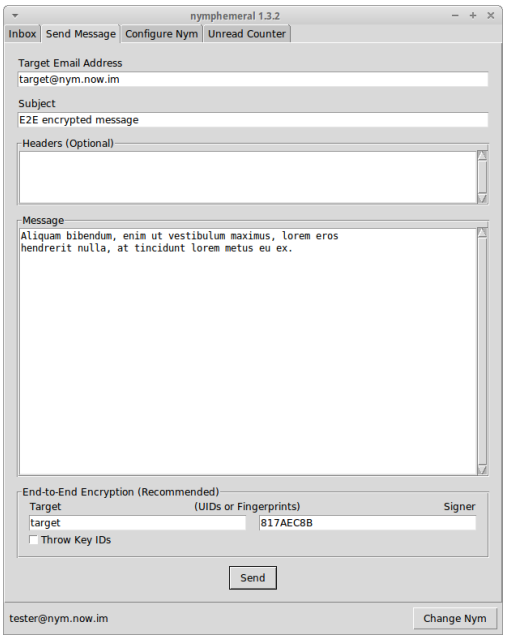

Fig. 3.12: End-to-End Encryption

The `End-to-End Encryption` section enables the user to encrypt and/or sign messages. The `Target` and `Signer` fields can receive either an UID or fingerprint. If more than one key is found for that query, nymphemeral will ask the user to be more specific, to avoid ambiguity.

There is also the option to use the `Throw Key IDs` checkbox, so that **if** someone obtains the ciphertext, they will not even be able to find out whom the message was encrypted to, because the key ID was removed. Therefore, when someone receives that message, they will have to use all of their keys to attempt to decrypt it.

If the message is being signed, either the GPG Agent or nymphemeral will prompt you for a passphrase to unlock the secret key:

Note: You should read the *GPG Keyring* section to add the keys involved in the End-to-End Encryption to nymphemeral's keyring.

3.6.3 Message Structure

It is important to know how the contents of your message are handled. For example, if you composed the following message:

```
To: recipient@domain
Subject: Foo

Bar
```

Fig. 3.13: GPG Agent

It would be encrypted to the server (with both asymmetric and ephemeral encryption layers) and would become the following message to be transmitted:

```
To: send@server

-----BEGIN PGP MESSAGE-----
<ciphertext>
-----END PGP MESSAGE-----
```

If someone intercepted the message, they would only learn that you sent a message to `send@server`, which would remail it to someone else. However, they learn nothing about the original message, because it is encrypted to the server. Now, when the server receives and decrypt it, the original message is accessed:

```
To: recipient@domain
Subject: Foo

Bar
```

That is the reason that another layer (end-to-end encryption) should be added. That way, when the server removes its encryption layer, it would only have access to the headers:

```
To: recipient@domain
Subject: Foo

-----BEGIN PGP MESSAGE-----
<ciphertext>
-----END PGP MESSAGE-----
```

It is called "end-to-end" because only the ends of the transmission (you and the recipient) can access the data. That last encryption layer must be removed by the recipient, to finally obtain the plaintext of the message. The last thing you should know is that the headers cannot be encrypted. Therefore, make sure to use non sensitive information for the **subject** and **optional headers** you might add.

3.7 Configuring the Nym

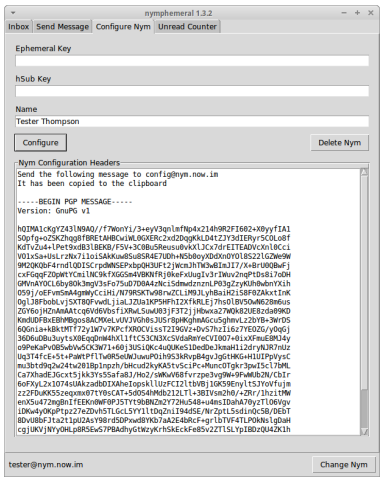

Fig. 3.14: Configure Nym Tab

On the `Configure Nym` tab you can modify the `Ephemeral Key`, `hSub key` and `Name` of your nym. Clicking on `Configure` will create the message that will be sent to the server. If you wish to not use your nym anymore, click `Delete Nym`.

Tip: If you believe that the server is not processing your messages, try to change the `Ephemeral Key` so that the databases will be synced.

3.8 Unread Messages Counter

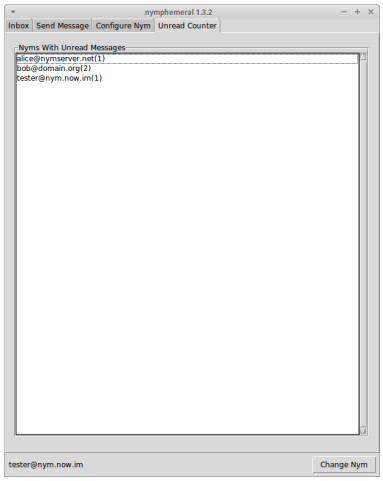

Fig. 3.15: Unread Counter Tab

On the `Unread Counter` tab there is a list of nyms with unread messages, so you do not need to log in with all your nyms to check if they have new messages (as long as you are retrieving messages with a nym that has access to the encrypted hSub passphrases).

CHAPTER

FOUR

OTHER

4.1 Changelog

4.1.1 nymphemeral 1.3.5, released 2015-09-09

- Improve the logger level
- Add more information to the installation sections
- Define which attributes and methods of Client are private
- Improve the code (add constants, simplify methods, remove redundancy)
- Fix bug to save new databases in the `db` directory

4.1.2 nymphemeral 1.3.4, released 2015-07-22

- Improve recognition of the Mixmaster installation
- Clarify nymphemeral's features and limitations
- Add instructions for Whonix

4.1.3 nymphemeral 1.3.3, released 2015-07-18

- Use pyaxo 0.4
- Use Python's logging module
- Slightly improve aampy's performance
- Improve parsing of the config file
- Improve and add more input validation
- Bug fixes and code improvements

4.1.4 nymphemeral 1.3.2, released 2015-05-13

- Add End-to-End encryption
  - Encrypt
  - Throw key IDs

  - Sign
  - Support GPG agent
- Redesign aampy

4.1.5 nymphemeral 1.3.1, released 2015-03-03

- Create client module
- Modify the GUI to be a layer between the user and the client

4.1.6 nymphemeral 1.2.3, released 2015-02-14

- Remove dependency links processing from *pip install*

4.1.7 nymphemeral 1.2.2.1, released 2014-11-14

- Remember the output method being used

4.1.8 nymphemeral 1.2.1, release 2014-11-10

- Append date to the title of the messages in the 'inbox'
- Encrypt hSub passphrases
- Support headers added by the user at the top of the message being composed
- Add the 'In-Reply-To' header to the reply

4.2 Feedback

Please report any suggestions, feature requests, bug reports, or annoyances to the GitHub issue tracker.

4.3 Acknowledgements

- Thanks to rxcomm for the new nymserv, nym.now.im, pyaxo, aampy and for assisting on the development of this client
- Thanks to crooks (Zax) for the original nymserv software
- Thanks to tych0 for assisting on fixes and improvements
- Thanks to HulaHoopWhonix (from the Whonix team) for testing and providing awesome feedback, such as bug reports, feature requests and suggestions