pEp MixMailer Documentation

Release 0.1

juga

Jan 29, 2021

Contents

CHAPTER 1

Analysis

1.1 Concepts

1.1.1 Anonymity

What is anonymity?

In the context of Privacy Enhancing Technologies (PET) [PET], what people usually mean by anonymity is **unlinkability**, which is still vague cause does not specify what with respect to who is unlinkable.

More specific terms related unlinkability, ie. who is talking to who with respect some adversaries with certain capabilities [AnonTerms].

- **sender anonymity**
- **receiver anonymity**
- **location anonymity**
- **third party anonymity**

Even more specific:

- **sender unobservability** : whether the sender is talking at all
- **receiver unobservability**

To be able to define the desired "properties" in "anonymous" communication systems, threat models should be specified.

1.1.2 Adversaries

Passive adversary

Adversary observing both ends

Can link sender and receiver by timing and volume patterns

Global Passive adversary

Active adversary

Adversary observing both ends

Confirmation attacks: Adversary can link sender and receiver by inducing timing signatures on the traffic to force distinct patters

Rogue operators

Malicious node operators. Passive or active.

1.1.3 Threat models

Which adversaries a system protect or does not protect against?

For instance, in Wikipedia edits:

- Sysadmins can link one user unregistered edit to another by the IP
- **If editing in a company, the company can see the amount of data at a** certain time which can match an be seen the public Wikipedia edit.

The following is based on [ApplicationThreatModeling] and [ThreatModelingOutputs]

Process

What are we building?

- architecture diagrams
- dataflow transitions
- data classifications

What can go wrong?

[STRIDE] and other structures can help.

Outputs

Template

Some data flow (eg. SMTP transmission of message from N1 to N2) Some data (eg. message being transmitted) Threat (eg. headers not encrypted, information disclosure), status (eg. open), severity (eg. high) Mitigation (eg. link encryption with TLS)

1.2 Requirements

TBD

Without defining the requirements first, any software we write will have to be re-written many times.

1.2.1 What to implement

```
Add mixnet capabilities to SMTP transport
```

1.2.2 What is it for? (updated Aug 18, 2020)

- to hide metadata, in this case, Email headers YES Email from the last node to the receiver will have the headers that pEp adds/removes/modifies
  - to a third party observer? (unlinkability, possible correlation attacks) YES
  - to the receiver? (sender anonymity) NO
    - what about the sender Email `From:` header when the receiver wish to reply to the sender?
    - what about the sender Email `Received:` header?, that might disclose sender geographic location?
    - what about the sender Email `Date:` header?, which discloses the time when the Email was sent
    - untraceable return address? (receiver can reply but doesn't know the location of the sender?)
  - to the sender? (receiver anonymity): NO the sender does not know which recipient has received the message.
- to be compatible with existing Mixmaster network? Mixmaster network still works, but small anonymity set, so NO
- to reuse the pieces that pEp or partners have developed? CAN
- to not to reinvent the wheel? SHOULD
- to have a system that anyone can extend? (scalability)
  - how to convince others to run nodes? > by the extra capabilities pEp offers > intial nodes can be run by friends

1.2.3 Who are users?

who is going to use MixMailer?

- pEp clients? YEA
- anyone? NO

estimated number of users/clients?:

```
pEp scales to billions of users
```

1.2.4 Who is going to operate the/maintain MixMailer?

- pEp company/foundation? NO
- pEp partners? NO
- anyone? YES
- estimated number of nodes?

1.2.5 Usability

It should be a solution that is usable by the "end user", not only by the nerd:

```
transparent to the user
```

1.2.6 Which would be the initial components of the system?

- pEp client (frontend), eg. Thunderbird plugin:

  ```
  pEp client is part of the engine
  ```

- pEp engine (backend), would route the Email to MixMailer

1.3 StateoftheArt

Analysis of the state of the art in the mix networks' systems and software.

1.3.1 Readings

[BibMixnets] contains all relevant papers about Mix networks.

The following papers are more relevant for Mixmailer:

- [Untraceable]
- [DesignAnonymous]
- [MixminionPaper]
- [Trickle]
- [Batching]
- [SurveyRouting]
- [MixCascadesP2P]
- [Trilemma]
- [Sphinx]
- [Loopix]
- [Stopandgo]

Recommended by D.:

- [Byzantine]
- [Uniform]
- [Gossip]

1.3.2 Onion routing

> Onion routing is a technique for anonymous communication over a computer network. In an onion network, messages are encapsulated in layers of encryption, analogous to layers of an onion. The encrypted data is transmitted through a series of network nodes called onion routers, each of which "peels" away a single layer, uncovering the data's next destination. When the final layer is decrypted, the message arrives at its destination. The sender remains anonymous because each intermediary knows only the location of the immediately preceding and following nodes

[OnionRouting]

1.3.3 Mix networks

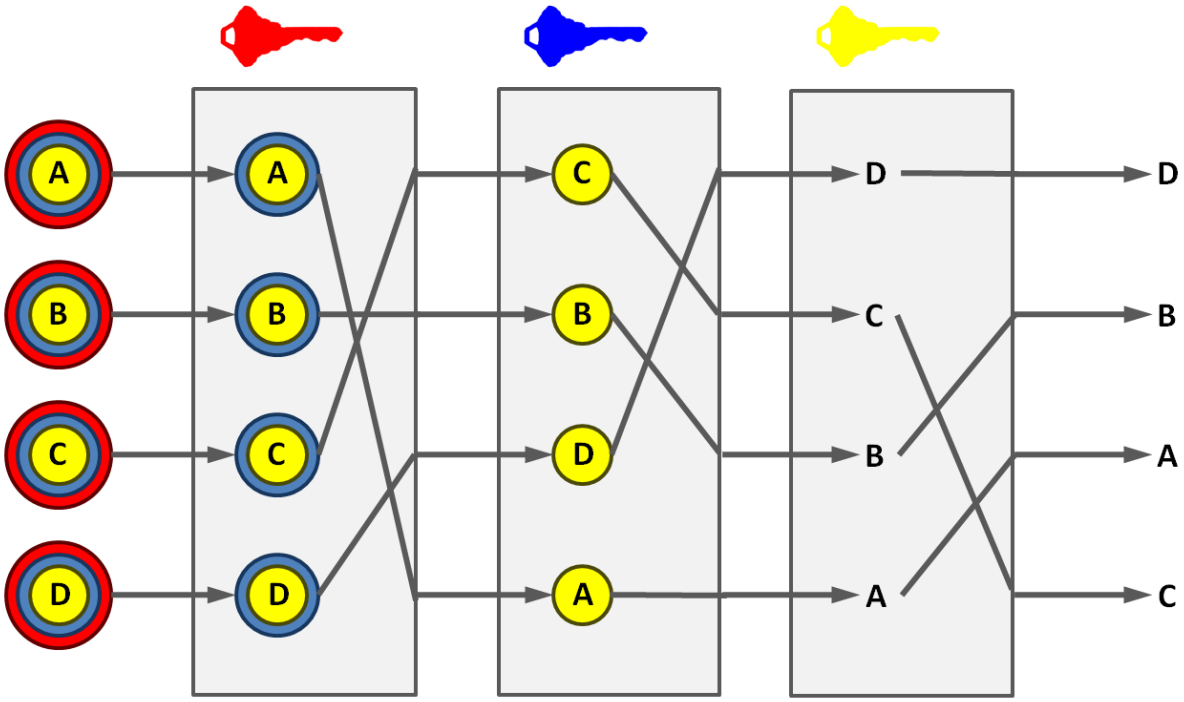
Mix networks systems and software considered to be used for the pEp MixMailer.

> I had used the terms "mix network" and "onion routing" almost interchangeably. In actuality I had fallen into a trap that a fair number of people familiar with the space have fallen into: using those terms without a solid differentiation. This blog post aims to correct that.

[MixOnion]

> Mix networks are routing protocols that create hard-to-trace communications by using a chain of proxy servers known as mixes which take in messages from multiple senders, shuffle them, and send them back out in random order to the next destination (possibly another mix node). This breaks the link between the source of the request and the destination, making it harder for eavesdroppers to trace end-to-end communications. Furthermore, mixes only know the node that it immediately received the message from, and the immediate destination to send the shuffled messages to, making the network resistant to malicious mix nodes.
>
> Each message is encrypted to each proxy using public key cryptography; the resulting encryption is layered like a Russian doll (except that each "doll" is of the same size) with the message as the innermost layer. Each proxy server strips off its own layer of encryption to reveal where to send the message next. If all but one of the proxy servers are compromised by the tracer, untraceability can still be achieved against some weaker adversaries.

[MixNetworks]



Pseudonymous remailers

A pseudonymous remailer or nym server, as opposed to an anonymous remailer, is an Internet software program designed to allow people to write pseudonymous messages on Usenet newsgroups and send pseudonymous email. Unlike purely anonymous remailers, it assigns its users a user name, and it keeps a database of instructions on how to return messages to the real user. These instructions usually involve the anonymous remailer network itself, thus protecting the true identity of the user.

[RemailerNym]

Nym servers

A nym server (short for "pseudonym server") is a server that provides an untraceable e-mail address, such that neither the nym server operator nor the operators of the remailers involved can discover which nym corresponds to which real identity.

To set up a nym, you create a PGP keypair and submit it to the nym server, along with instructions (called a reply block) to anonymous remailers (such as Cypherpunk or Mixmaster) on how to send a message to your real address. The nym server returns a confirmation through this reply block. You then send a message to the address in the confirmation.

[NymServer]

Anonymous remailers

An anonymous remailer is a server that receives messages with embedded instructions on where to send them next, and that forwards them without revealing where they originally came

from. There are Cypherpunk anonymous remailers, Mixmaster anonymous remailers, and nym servers, among others, which differ in how they work, in the policies they adopt, and in the type of attack on anonymity of e-mail they can (or are intended to) resist

[AnonymousRemailer]

Type I (Cypherpunks)

A Cypherpunk remailer sends the message to the recipient, stripping away the sender address on it. One can not answer a message sent via a Cypherpunk remailer. The message sent to the remailer can usually be encrypted, and the remailer will decrypt it and send it to the recipient address hidden inside the encrypted message. In addition, it is possible to chain two or three remailers, so that each remailer can't know who is sending a message to whom. Cypherpunk remailers do not keep logs of transactions.

[TypeI]

More documentation on Cypherpunk remailers:

https://www.iusmentis.com/technology/remailers/index-cpunk.html

https://www.iusmentis.com/technology/remailers/index-anon.html

Last modified: 20 Jun 1998

Properties:

- Transport (between the Cypherpunk remailers): SMTP / POP3
- Format: MIME?

Cons:

- replay attacks
- flooding attacks
- **no central directory server, which creates an asymmetry of** information about the remailers in the network. The not equally distributed knowledge of current servers can be exploited to undermine anonymity and is one of the Cons of the system

  [HuMixmaster]
- no user friendly clients
- old crypto
- no possibility to replay
- no network diversity/poor anonymity set
- spam

Type II Mixmaster

In Mixmaster, the user composes an email to a remailer, which is relayed through each node in the network using SMTP, until it finally arrives at the final recipient. Mixmaster can only send emails one way. An email is sent anonymously to an individual, but for them to be able to respond, a reply address must be included in the body of the email. Also, Mixmaster remailers

> require the use of a computer program to write messages. Such programs are not supplied as a standard part of most operating systems or mail management systems.
>
> [TypeII]

Mixmaster Protocol Version 2, December 29, 2004:

Cryptographic Algorithms:

> The asymmetric encryption mechanism is RSA with 1024 bit RSA keys and the PKCS #[MixOnion] v1.5 (RSAES-PKCS1-v1_5) padding format [Nymtech]. The symmetric encryption mechanism is EDE 3DES with cipher block chaining (24-byte key, 8-byte initialization vector). MD5 is used as the message digest algorithm.

Packet format:

> The header sections (except for the first one) and the packet body are encrypted with symmetric session keys
>
> [Mixmaster]

Properties:

- topology: cascade (no scalability, then poor anonymity set)
- **mixing strategy: batch and reorder (unpredictable delays) Have an** adjustable size message pool in which incoming packets are collected and re-sorted randomly before being forwarded.
- Key/node discovery: pinger
- Transport: SMTP
- Format: MIME?
- Packets: all packets are the same size.
- Packet format:
- Cover traffic: can send dummy packages
- Compatible with Cypherpunk messages.

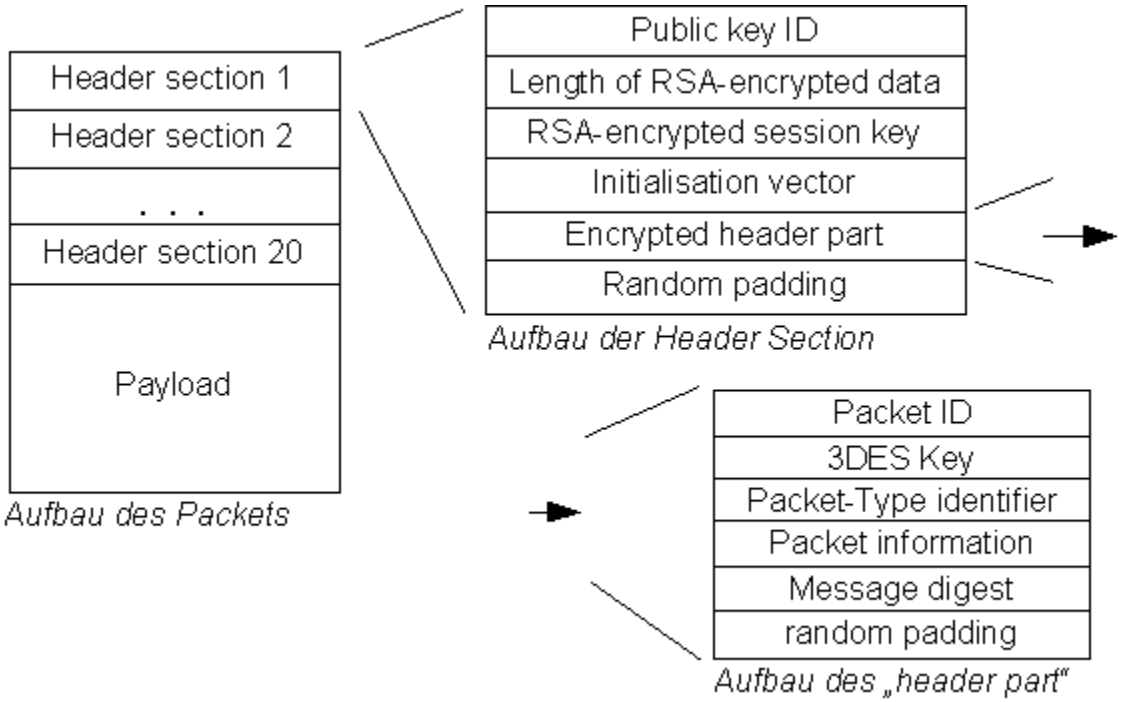
Pros:

- third party untraceability between sender and receiver
- resistant to timing correlation attacks
- **support replies or anonymous recipients (not resistant to replay** attacks)

Cons:

- not resistant to active attacks, eg. dropping packets and causing denial of service
- not resistant to replay attacks
- no central directory server (network partitioning)
- no user friendly clients
- old crypto
- no replies
- no network diversity/poor anonymity set
- spam

Packet structure:



Type III Mixminion

> A Mixminion remailer attempts to address the following challenges in Mixmaster remailers: replies, forward anonymity, replay prevention and key rotation, exit policies, integrated directory servers and dummy traffic. They are currently available for the Linux and Windows platforms. Some implementations are open source.

[TypeIII]

> 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.

[Mixminion]

Properties:

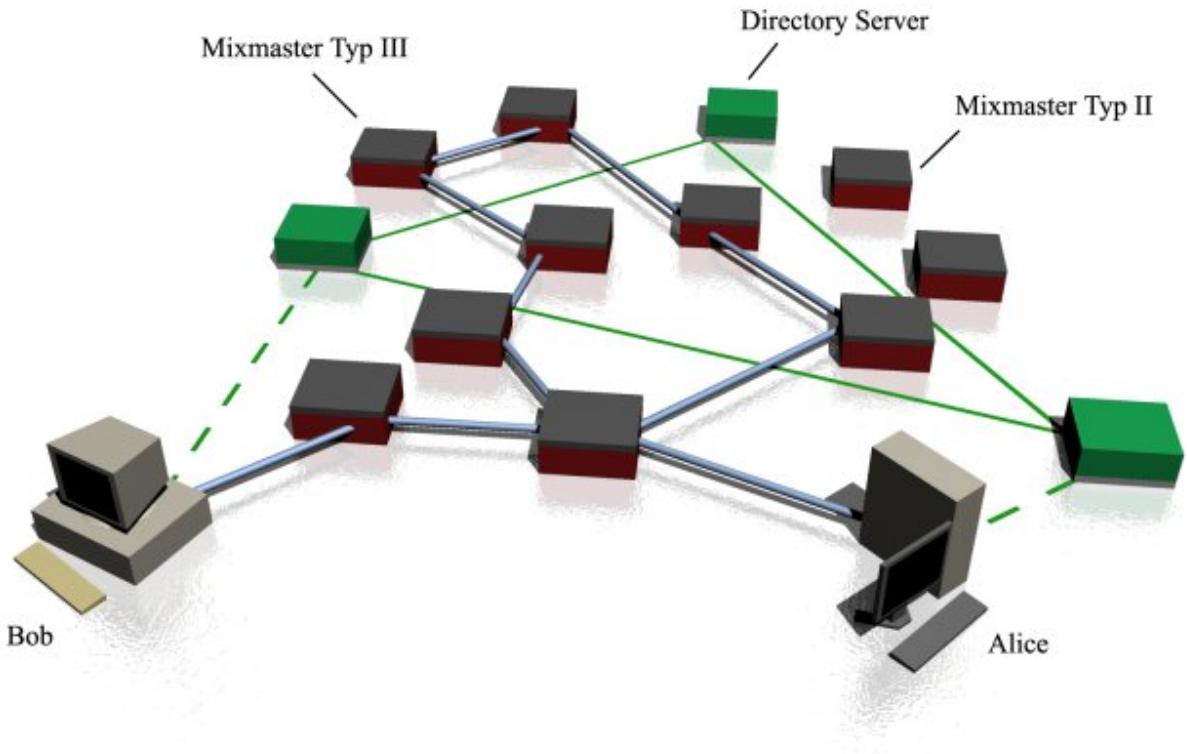
- Transport: TLS instead of SMTP.
- Key/node discovery: directory servers or pinger
- Key rotation:
- Up to 32 instead of 20 hops.
- Packet format: Sphinx
- Compatible with Cypherpunk and Mixmaster messages.

Pros:

- directory servers (no partitioning)
- Indistinguishable between forward and reply messages.
- Forward secrecy by using TLS

Cons:

- no user friendly clients
- old crypto
- no network diversity
- spam



Remailer vulnerabilities

https://www.freehaven.net/anonbib/cache/rprocess.html

Katzenpost

Properties:

- topology: stratified (scales, better anonymity set)
- mixing strategy: stop and go (predictable relays)

- packet format: Sphinx
- **it has Providers as entry point that require authentication,** therefore it is needed to first
- sender needs to know the receiver Provider (no receiver anonymity?)

Pros:

- third party untraceability between sender and receiver
- better scalability, therefore better network diversity/anonymity set
- predictable delays
- **it could be possible to use the mixnet to send Email only between two** organizations that "trust" each other without SPF to have 3rd party unlinkability

Cons:

- **License is AGPL v3, not compatible with xGPL v2, which means it can't** be part of iOS. katzenpost license can not be changed cause one of their authors. We'd need to rewrite katzenpost client and mailproxy to use it in iOS
- **not designed to run with Email messages there is a mailproxy, but** katzenpost devs are unhappy with the mail proxy
- no production network atm
- sender needs to authenticate with a Provider
- sender needs to know the receiver Provider (no sender anonymity)
- **for the providers to be able to send Email to any destination, they** need to use SPF, therefore no sender anonymity
- **different Email traffic patterns (eg. Deltachat) would need different** delay parameters

  [katzenpostSite]

Nym mixnet

Cons:

- **License is Apache, not compatible with GPL. So far all software** created by pEp must be GPL(-compatible).

  [Nymtech]

1.3.4 Mix networks software

reliable

Remailers Type I and II users' maunal :

https://gateway.ipfs.io/ipfs/QmboFHizh9ys57DXcVsniVDYS46gsiBP716u2sqQE7xgV4

https://www.panta-rhei.dyndns.org/JBNR-en.htm#CForm

Revised: 23-Sep-99 [Reliable v1.0.2]

Are the OpenPGP encrypted emails not padded?

mixmaster

Code in sourceforge [^mixmaster origin] is not in CVS system.

Uses RSA (and DSA?) for encryption and PKCS#1 for padding.

Several repositories with some updates

https://github.com/eurovibes/mixmaster:

> Mixmaster version 3 – anonymous remailer software – (C) 1999 Anonymizer Inc.

Commits from Nov 2001 to Oct 2001?

One of the first versions, 2.9beta32 says:

> It supports OpenPGP encryption (compatible with PGP 2, PGP 5 and GnuPG).

(Only for the recipient or also for the mixes?)

https://github.com/crooks/mixmaster

Commits from 2004 (including previous code) to 2014

https://github.com/merkinmuffley/mixmaster4096

Commits from 2014 to 2018

Encryption with 4096 bits RSA

https://github.com/crooks/pymaster

Mixmaster version 3 implementation in Python

Commits from Feb 2013 to Jun 2013.

Uses RSA for encryption and PKCS#1 for padding.

OpenPGP is used only for the remailer admins.

No license

https://github.com/crooks/mimix

> provide a Mixmaster-like protocol with support for larger RSA keys (up to 4096 bits), replacement Hash (SHA2) and Symmetric Ciphers (AES). Instead of using SMTP for inter-Remailer traffic, Mimix uses HTTP.

Commits from Oct 2013 to Mar 2014

GPL license

Mixminion

Commit from 2002 to 2012.

C bindings and Python

License until v1.0.0 GPL, later BSD

https://github.com/tim54000/cypherpunk-cli

Rust

2019

https://github.com/byte-mug/go-cypherpunk

Go

Aug 2020

Running remailers

https://apricot.fruiti.org/echolot/rlist2.html

Last update: Thu Sep 3 10:30:01 2020.

All OpenPGP remailer keys are dsa1024 for signing and elg1024 for encryption, except for remailer@kroken.de.eu.org, which are rsa4096.

```
$remailer{"austria"} = "<mixmaster@remailer.privacy.at> cpunk max mix pgp pgponly␣
↪repgp remix latent hash cut test ekx inflt50 rhop5 reord klen1024";
$remailer{"cloaked"} = "<mixmaster@cloaked.pw> cpunk max mix middle pgp pgponly repgp␣
↪remix esubbf hsub latent hash cut test ekx inflt50 rhop5 reord post klen1024";
$remailer{"dizum"} = "<remailer@dizum.com> cpunk max mix pgp pgponly repgp remix␣
↪latent hash cut test ek ekx esub inflt50 rhop5 reord post klen64";
$remailer{"frell"} = "<godot@remailer.frell.eu.org> cpunk max mix pgp pgponly repgp␣
↪remix latent hash cut test ekx inflt50 rhop5 reord post klen1024";
$remailer{"hsub"} = "<hsub@mixmaster.mixmin.net> cpunk max pgp pgponly repgp remix␣
↪esubbf hsub latent hash cut test ekx inflt50 rhop5 reord klen100";
$remailer{"kroken"} = "<remailer@kroken.de.eu.org> cpunk max mix middle pgp pgponly␣
↪repgp remix latent hash cut test ekx inflt50 rhop5 reord klen1024";
$remailer{"lambton"} = "<remailer@lambton.org> cpunk max mix middle pgp pgponly repgp␣
↪remix esubbf hsub latent hash cut test ekx inflt50 rhop5 reord klen256";
$remailer{"paranoia"} = "<mixmaster@remailer.paranoici.org> cpunk max mix pgp pgponly␣
↪repgp remix latent hash cut test ekx inflt50 rhop5 reord post klen150";
$remailer{"redjohn"} = "<remailer@redjohn.net> cpunk max mix middle pgp repgp remix␣
↪esub hsub latent hash cut test ek ekx inflt50 rhop5 reord post klen1024";
$remailer{"senshi"} = "<senshiremailer@gmx.de> cpunk middle pgp latent ek ekx esub␣
↪cut hash repgp reord ext max test inflt10 rhop2 klen200";
$remailer{"zip2"} = "<mix@mail.zip2.in> cpunk max mix pgp pgponly repgp remix esubbf␣
↪hsub latent hash cut test ekx inflt50 rhop5 reord post klen1024";
```

Katzenpost

See *Katzenpost*

Nym Mixnet

See *Nym mixnet*

1.3.5 Comparative remailer mix neworks, onion routing and pEp

This is not an exhautive comparative and might not be accurate.

It is base on the pages:

- *Type I (Cypherpunks)*
- *Type II Mixmaster*
- *Type III Mixminion*
- *Onion routing*
- *Proposal 1 (P1) design*

	Remailer	Mixnetwor ks		Onion Routing	pEp MixMailer
	Type I/Cypherp unks	Type II/Mixmas ter	Type III/Mixmi nion	Tor	
transport	SMTP	SMTP	TLS + SMTP?	TLS	SMTP?
nodes key	NA	yes	yes	yes	yes (OpenPGP) ?
key rotation	NA	no	yes	yes	active/pa sive
nodes/key discovery	manual	pinger?	directory	dirauths	GNS?
types of nodes	NA	same?	several?	"G,M,E"	same?
packet format	?	?	Sphinx	specific	?
packet size	different	same?	same	same	different in fu- ture?
packet mixing	NA	yes	yes	no	yes in future?
cover traffic	NA	yes	yes?	no	?
directory server	no	no	central	decentral ized	GNS/trust ed nodes?
reply	no	no	SURB	"yes, cir- cuit"	NA
spam	yes	yes	yes	NA	?
network diver- sity	no	no	no	yes	?
user friendly	no	no	no	yes	?
old crypto	yes	yes	yes	no	?

1.3.6 "Open" vs "close" system

In both *Proposal 1 (P1) design* and *Proposal 2 (P2)* we should decide whether it should be possible to send Email to any Email service, eg. gmail or to a few Email services, eg. riseup and systemli or several different companies.

In the "open" system there's no solution to avoid spam. If the public keys are available and anyone can join the mixnet, spammers can also send encrypted spam. Also, services like Gmail will refuse to receive Email from other domains/services not using [SPF]. If the mixnet provider/service uses SPF, then it's not possible to have sender anonymity (unlinkability).

In the "close" system, a malicious provider could also inject spam, but it'd be less likely, specially if the users have to authenticate to the providers, which is the case of *Katzenpost* .

Diagram with errors:

Fig. 1: image

Other

1.3.7 Peer to Peer networks

GNUnet

> GNUnet is a software framework for decentralized, peer-to-peer networking and an official GNU package. The framework offers link encryption, peer discovery, resource allocation, communication over many transports (such as TCP, UDP, HTTP, HTTPS, WLAN and Bluetooth) and various basic peer-to-peer algorithms for routing, multicast and network size estimation.
>
> GNUnet's basic network topology is that of a mesh network. GNUnet includes a distributed hash table (DHT) which is a randomized variant of Kademlia that can still efficiently route in small-world networks. GNUnet offers a "F2F topology" option for restricting connections to only the users' trusted friends. The users' friends' own friends (and so on) can then indirectly exchange files with the users' computer, never using its IP address directly.
>
> GNUnet uses Uniform resource identifiers (not approved by IANA, although an application has been made).[when?] GNUnet URIs consist of two major parts: the module and the module specific identifier. A GNUnet URI is of form gnunet://module/identifier where module is the module name and identifier is a module specific string.

[gnunetGNUnet]

> > A distributed hash table (DHT) is a distributed system that provides a lookup service similar to a hash table[GNUnet]: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key.
>
> [DHT]

GNS

> GNUnet includes an implementation of the GNU Name System (GNS), a decentralized and censorship-resistant replacement for DNS. In GNS, each user manages their own zones and can delegate subdomains to zones managed by other users. Lookups of records defined by other users are performed using GNUnet's DHT

[gnunetGNS]

1.4 Attacks

Based on [AttacksAnonymitySystems] talk on 2003 that explained the design of *Type III Mixminion* to mitigate the vulnerabilities the previous remailers: *Type II Mixmaster* and *Type I (Cypherpunks)*.

Explanation of the attack, goal of the attack, which are the adversaries, which are the capabilities or the adversarios and which the mitigations (M)

1.4.1 Distinguish different-size messages

A passive local adversary can guess which message coming out of a mix corresponds to a message coming in just by the size. They can guess also the position of the message in the path.

Mitigation: padding. Add random junk to the bottom to replace the header that is removed.

Cyhperpunk remailer is vulnerable to this, but not mixmaster or mixminion.

1.4.2 Replay attack

An adversary copy the message and send it back, it will follow the same path. It's traceable by a global passive adversary. It helps local passive adversaries and rogue operators. Depends on where the message is going, eg. same ISP.

Mitigation: add hash to the header and get nodes to cache the hashes. If an incoming message is already in the cache, drop it.

Problem: nodes would need to remember cache forever.

Mitigation: add expiration date to the cache, eg. 3 days ago and 3 days from now. If expiration of cache is more than 7 days old, drop it. Adversaries can't tell when the message was sent from its expiration date.

Cyhperpunk remailer is vulnerable to this attack, but not mixmaster or mixminion.

1.4.3 Flooding (Blending) attack

An active local adversary send many messages to detect a legit one. Very effective attack in batch mixes, but not in pooling ones. There're other cheaper and more effective attacks.

Mitigation: pooling mixnet.

1.4.4 Trickle attack (n-1)

If there's only one user (1 legit message) at a given interval of time, flooding attack is also effective in pooling mixes.

Mitigation: dummy decoy messages.

Problem: dropped at some node, only covers internal traffic.

Cyhperpunk remailer is vulnerable to this attack, but not mixmaster or mixminion.

1.4.5 Passive subpoena attack

A local passive adversary copy message for a later subpoena.

Mitigation: encrypt link transmission (TLS)

Problem: TLS can be enforced in nodes running TLS and postfix doesn't rotate keys too often.

Solution: enforce link transmission encryption at protocol level and have effemeral keys that rotate often.

Cyhperpunk and mixmaster remailers are vulnerable to this attack, but not mixminion.

1.4.6 Active subpoena attack

A rogue operator can record messages for a later subpoena.

Mitigation: periodic key rotation.

Cyhperpunk and mixmaster remailers are vulnerable to this attack, but not mixminion.

1.4.7 Partition attack on client knowledge

Adversary make some users to connect to a different set of nodes that it's easier to observe or control by the adversary.

Different versions of OpenPGP encrypt differently and looks different. Also choosing different algorithms. Watching users that are using a key about to expire (not that common).

It needs the atacker to do something else. It can be statistical analysis over time.

Mitigation: Directory servers (DSs) have to be part of the specification to avoid different DSs behaving differently. They also have to sign bundles. Problem: how can DSs agree on what to sign?

Question: having DSs is not P2P? Answer: there're several DSs that talk to each other, so it's P2P in that sense But there's an small number of DSs and it's static cause their public keys need to be in the code.

1.4.8 Partition attack on message expiration date

The adversary delays a message near to its expiration date, so that they can recognize it cause because there're not that many. Statistical attack, not deterministic.

Mitigation: No expiration dates, keep hashes until keys rotate.

1.4.9 Tagging attack on headers

An adversary flips bits in the encrypted payload at the possition of the header. If the adversary owns a node later in the path that correspond to the broken header, they can recognize the message.

Mitigation: hash covering the entire header.

Mixmaster is vulnerable to this attack. There're easier and more effective attacks.

1.4.10 Tagging attack on payload

Adversary flips some bits in the payload, and try to recognize altered messages when they're delivered.

Mitigation: Make the hash cover the payload too.

1.4.11 Attack on multiple messages

Large messages that are splitted into packets, they can have different routes, but they all need to end in the same last remailer to reconstruct the message.

A passive global adversary observing only input and output of the mixnet can correlate the messages by number of them. Users can only "hide" with other users using the net in the same way, eg. messages containing a vote or pictures have different patterns. Also passive local adversaries can gather enough information observing the net enough time.

Very effective attack: don't send large files unless every body else does it too.

Cyhperpunk, mixmaster and mixminion are vulnerable to this attack.

1.4.12 Pseudospoofing

Rogue operator running several nodes. In mixmaster an operator can join the network with human interaction, though the operator can have different personas and operate large part of the network. In mixminion there's no human interaction. Tor uses "families", scripts, authorities and humans to detect these cases. Ratio of users per node, not clear If user retrieves the nodes they are going to use by pinging them, then it's lear which nodes they're going to use.

1.4.13 Intersection attacks

Adversaries can know who are the users over time, cause the tend to do the same thing more than once. Over time it's possible to know who they are. eg. every morning a user sends message to same receiver, except when they are on holidays. Difficult except for a global passive adversary or rogue operator that happens to be consistenly the 1st and last hop.

1.4.14 Timing and packet counting attacks

Adversary do statistical analysis of network traffic. Mostly on low-latency systems with pool of messages.

Mitigation: padding. Even if timing analysis is harder with delays, the system can not add too much delay to keep it usable.

1.4.15 Bottleneck compromises

In a nym server system obviously the nym server is a Bottleneck since all the traffic goes throug it.

A node is more reliable might be used more often and therefore attacking/owning easies attacks. Trade-off between having users choose reliable servers or servers in one jurisdiction.

CHAPTER 2

Design

2.1 Proposal 1 (P1)

2.1.1 Code API / Interface

Proposed by V.

Node Interface to register as a node

```
PEP_STATUS register_node(PEP_SESSION session, pEp_identity *ident);
```

```
def register_node(ident):
    "parameter ident must be a pEp.Identity"
```

Client Interface to discover nodes

```
PEP_STATUS get_registered_nodes(PEP_SESSION session, identity_list **nodes);
```

```
def get_registered_nodes():
   "returns list with identities"
```

2.1.2 Proposal 1 (P1) design

aka pEp onion routing or mix network or remailer?

TBD.

Based on notes talking with V. on February 3, 2020 and in March.

See *Comparative remailer mix neworks, onion routing and pEp* for an overview about onion routing and mix networks.

Routing

Client decide the route for the message. > The client encrypts the message with the recipients OpenPGP key and some nodes' OpenPGP keys.

See *pEp mesages structure*.

This in a similar to *Type I (Cypherpunks)* remailers.

> The client sends the message to the first node. The first node receive the message and decrypt it. It can know whether the message is from a trusted node (by the color).

(How the color is calculated?, by the signature of the sender?)

Since the message is already encrypted for the next node, there is no need to re-encrypt it.

(Does the node needs to sign it before sending it to the next node?)

Transport

> SMTP. No plan to enforce TLS cause Sequoia is working on Forward Secrecy.

(How would forward secrecy work with different layers of encryption?)

Nodes' keys

> It would be the same type of key that the engine is using. If engine is using Sequoia, the keys will be OpenPGP. It could be symetric keys, but for onion routing they need to be asymmetric.

(Not really true. Tor generates a symetric session key for circuit to minimize the expensive computation with asymetric keys)

Key rotation

> No automatic key rotation. The engine do key distribution and key reset Key reset can happen actively or passively.

(In Katzenpost key rotation happen often and the key rotation moment is vulnerable to attacks. OpenPGP keys don't change that often.)

Key discovery (by clients)

> The first time that a user/node send a message, the key is attached. A client needs to exchange a couple of messages with every node to discover their keys.

(How would this scale with thousands of nodes?)

Nodes and key discovery by clients and nodes

Via *GNS*

> Nodes should register, with an email address and a key, what is already in pEp identity. The key should be in ASCII armor Clients obtain the nodes from GNS.

(did V. mention Bézier curve?)

CG proposed to use the CERT record type [CERT]

Routing

> First client obtain the nodes. Then client choose the route It is still needed to define the algorithm to choose the route. Initially it could be a dummy algorithm, like choose 3 nodes randomly. There should be a maximum of hops to choose.

Even choosing always the same route path, some degree of anonymity can be achieved by mixing packets of the same size and introducing delays.

Notes on Aug 12, 2020:

> In the context of Mixnets "free routing" gives actually less anonymity since there's less entropy, so it's actually better to implement cascade topology (fixed position in the route). With this topology as long as 1 node is honest, it's fine. It's even better, to have an stratified topology, in which there're groups of nodes for then entry, the middle and the exit. The problem of free routing is that eventually you might end up choosen a path where all the nodes are compromised. In Tor this is minimized by having guard (sticky entry) nodes.

Types of nodes

> All the same. This avoid the exit node problems.

(which exit node problems avoid this?, spam?)

Packet format

> Messages are MIME Multipart Encrypted (and Signed?) So far messages are not divided in packets nor padded. The message encrypted for all layers would need to be divided into packets. This is still not implemented. Clients might need that a message is a packet for debugging.

Packet size

> So far packets would have the size that the underlying transport give it to them. Once the engine divide messages in packets, they can be padded. The engine is already capable to pad messages.

(packets need to be all the same size -and mixed- to achieve unlinkability)

Notes on Aug 12, 2020:

> Impl. fixed-sized padding to messages. We could have 2 different packet/message size. One "large" and one "small". eg. 50mb and 50kb)

OpenPGP already does padding and we would need to modify that operation, but we should not modify the way OpenPGP works.

Only remailers Type I use OpenPGP for encryption.

Packet mixing

> There would be mixing of packets once they're the same size.

(If there is no packet mixing, it won't be a mix network nor a remailer.)

Cover traffic

Not planned

Notes on Aug 12, 2020:

Cover traffic mitigates statistical disclosal attack/analysis, which adds the "unobservability" property.

Delays

We have not discussed this.

Notes on Aug 12, 2020:

Delay and (one of the strategies, mixmaster's one, is "pooling") (could use postfix queue), which add the "unlinkability" property

Directory server

There are not directory servers. Trusted nodes using trustwords.

(How this would work?)

Reply

Since the recipient sees who was the original sender, there is no need for special type of reply.

(Then there won't be sender anonymity from the the recipient point of view)

Spam

Since the Emails are encrypted, there won't be spam.

Remailers require to have the node key to send an Email to them, that did not prevent spam.

(How would spam be prevented using discoverable OpenPGP keys?, how would the Emails arrive to recipients without SPF or DKIM?, how there could be sender anonymity with SPF?)

Network diversity

Initially there would be only pEp nodes.

(anonymity company, how other people would be incentivated to run nodes?)

Crypto

Modern OpenPGP crypto

(OpenPGP computation might be expensive?)

Deployment diagram

Deployment

2.1.3 pEp mesages structure

pEp messages structure when encrypting/decrypting several layers.

Example in which Alice encrypts first a message for Bob, then encrypts the encrypted message to Carol. Alice sends the message to Carol. Carol receives de message, decrypt, and sends the body of the decrypted message to Bob. Bob receives, decrypt it and see the message that Alice sent. The communication happens then in this direction:

Alice -> Carol -> Bob

Alice wants to send this message to Bob

```
Hi Bob,
this is Alice!
```

Alice creates the message for Bob

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="4bc216fb60fb5f2d6a48d71f5810ff9e"

--4bc216fb60fb5f2d6a48d71f5810ff9e
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

Subject: Subject: from Alice to Bob

Hi Bob,
this is Alice!

--4bc216fb60fb5f2d6a48d71f5810ff9e
Content-Type: application/pgp-keys
Content-Disposition: attachment; filename="pEpkey.asc"

-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----

--4bc216fb60fb5f2d6a48d71f5810ff9e--
```

Alice encrypts the message for Bob

```
From: Alice Lovelace <alice@openpgp.example>
To: Bob Babagge <bob@openpgp.example>
Subject: =?utf-8?Q?p=E2=89=A1p?=
X-pEp-Version: 2.1
MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary="2cfc2f5be8c232219b7b77f1a79092a";
protocol="application/pgp-encrypted"

--2cfc2f5be8c232219b7b77f1a79092a
Content-Type: application/pgp-encrypted

```

(continues on next page)

(continued from previous page)

```
Version: 1
--2cfc2f5be8c232219b7b77f1a79092a
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="msg.asc"

-----BEGIN PGP MESSAGE-----
[...]
-----END PGP MESSAGE-----

--2cfc2f5be8c232219b7b77f1a79092a--
```

Alice creates a message for Carol

Note that the message headers from Alice to Bob are included in the body of the message from Alice to Carol.

```
From: Alice Lovelace <alice@openpgp.example>
To: Carol Hopper <carol@openpgp.example>
Subject: Subject: from Alice to Bob
MIME-Version: 1.0
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

=46rom: Alice Lovelace <alice=40openpgp.example>
To: Bob Babagge <bob=40openpgp.example>
Subject: =3D=3Futf-8=3FQ=3Fp=3DE2=3D89=3DA1p=3F=3D
X-pEp-Version: 2.1
MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary=3D=2256788e955154139fffab44e5=
f290f91=22; =20
protocol=3D=22application/pgp-encrypted=22

--56788e955154139fffab44e5f290f91
Content-Type: application/pgp-encrypted

Version: 1
--56788e955154139fffab44e5f290f91
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename=3D=22msg.asc=22

-----BEGIN PGP MESSAGE-----
[...]
-----END PGP MESSAGE-----

--56788e955154139fffab44e5f290f91--
```

Alice encrypts the message for Carol

```
From: Alice Lovelace <alice@openpgp.example>
To: Carol Hopper <carol@openpgp.example>
Subject: =?utf-8?Q?p=E2=89=A1p?=
```

(continues on next page)

(continued from previous page)

```
X-pEp-Version: 2.1
MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary="6fa56ec33679af179f5f59a10722e17";
protocol="application/pgp-encrypted"

--6fa56ec33679af179f5f59a10722e17
Content-Type: application/pgp-encrypted

Version: 1
--6fa56ec33679af179f5f59a10722e17
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="msg.asc"

-----BEGIN PGP MESSAGE-----
[...]
-----END PGP MESSAGE-----

--6fa56ec33679af179f5f59a10722e17--
```

and sends it to Carol

Carol receives the message and decrypts it

```
From: Alice Lovelace <alice@openpgp.example>
To: Carol Hopper <carol@openpgp.example>
Subject: Subject: from Alice to Bob
X-pEp-Version: 2.1
X-EncStatus: reliable
X-KeyList:
EB85BB5FA33A75E15E944E63F231550C4F47E38E,EB85BB5FA33A75E15E944E63F231550C4F47E38E,
↪37D13DE1DCBAFCE7BCE117EE311399EB28E3E8AA
MIME-Version: 1.0
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

=46rom: Alice Lovelace <alice=40openpgp.example>
To: Bob Babagge <bob=40openpgp.example>
Subject: =3D=3Futf-8=3FQ=3Fp=3DE2=3D89=3DA1p=3F=3D
X-pEp-Version: 2.1
MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary=3D=225a2e1b238a9435439d7f32eb=
295b6f=22;  =20
protocol=3D=22application/pgp-encrypted=22

--5a2e1b238a9435439d7f32eb295b6f
Content-Type: application/pgp-encrypted

Version: 1
--5a2e1b238a9435439d7f32eb295b6f
Content-Type: application/octet-stream
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename=3D=22msg.asc=22
```

(continues on next page)

(continued from previous page)

```
-----BEGIN PGP MESSAGE-----
[...]
-----END PGP MESSAGE-----

--5a2e1b238a9435439d7f32eb295b6f--
```

Carol sends the body of the message to Bob

Bob receives and decrypt it

```
From: Alice Lovelace <alice@openpgp.example>
To: Bob Babagge <bob@openpgp.example>
Subject: Subject: from Alice to Bob
X-pEp-Version: 2.1
X-EncStatus: reliable
X-KeyList:
EB85BB5FA33A75E15E944E63F231550C4F47E38E,EB85BB5FA33A75E15E944E63F231550C4F47E38E,
↪D1A66E1A23B182C9980F788CFBFCC82A015E7330
MIME-Version: 1.0
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="msg.txt"

Hi Bob,
this is Alice=21
```

2.2 Proposal 1.1 (P11)

aka pEp onion routing/remailer?

This is an attempt to transcribe V. ideas after juga's presentation on January 12, 2021 [MixmailerSlides] and some other comments with N. and D. These are not juga's ideas.

As D. noted, it should probably not be called mixnet.

This proposal very similar to *Proposal 1 (P1) design*. The main difference seems to be not aiming to implement a mix network but just onion routing, which it is actually simpler.

Any mention to prototype means this: [Mixmailer]

2.2.1 Routing

As in Proposal 1 (P1), client decide the route for the message.

The user might also decide.

2.2.2 Transport

As in P1. SMTP, no TLS.

2.2.3 Nodes' keys

Same as in P1. Each node have its own private/public OpenPGP keys.

2.2.4 Keys rotation

Not defined

2.2.5 Key discovery (by clients and nodes)

In P1 it was proposed that nodes/clients exchange messages to discover keys. Then we started to use GNUnet *GNS* in the prototype. In this proposal we'll continue with GNS is used to query the list of nodes to the authorities and other nodes' keys. In the prototype it's explained in more detail: Mixnet nodes registration in GNS, though the part about layers should be ignored.

2.2.6 Key registration

This is being researched by N.

2.2.7 Types of nodes

All the same. In the prototype juga decided thought it would be better to use stratified topology, but it can be removed.

2.2.8 Topology

Free-routing.

2.2.9 Packet format

No packets, just messages. Not really MIME Multipart Encrypted (and Signed?). pEp message 2.x? [ref?].

2.2.10 Packet size

Instead of padding in a way that the final messages are all the same few fixed sizes, they'll be padded randomly. If the final message is too big it'll be just discarded. This is different to what was tried to implement in the prototype and P1.

2.2.11 Packet mixing & delays

Undecided. If it's just onion routing, in principle there's no need for. In the prototype there was just random delays.

2.2.12 Cover traffic

None.

2.2.13 Directory servers/authorities

In P1 it was said that there was no need. Since we started to use GNS, they're needed. The prototype already assume there'll be authorities.

2.2.14 Spam

It doesn't exist.

2.2.15 Network diversity

Community will be made.

2.2.16 Nodes and messages diagram

Nodes and messages

2.2.17 Deployment diagram

As the one before, including software components.

Deployment

2.2.18 Deployment diagram including GNS

As the one before, including GNS.

Deployment with GNS

See *Proposals 1, 2 and 1.1 arguments* for arguments in pro/con of the several proposals.

2.3 Proposal 2 (P2)

Katzenpost + [Mailproxy]

Mailproxy is a mailserver & mixnet client intended to run on the users' devices. Not as a provider's mail server, as this would create an "entry-node" to the mixnet which defeats certain security guarantees.

The client would need to configure the SMTP socket address (IP:port) of the mailproxy.

Outlook clients use [MAPI] . It's possible to use MAPI over HTTP.

2.3.1 Differences between loopix and katzenpost

In loopix there's only 3rd party anonymity (unlinkability), while katzenpost introduced the concept on "ephemeral" providers, which are the entry point for the client/user to query their "remote spool" to retrieve/send the messages via Single Use Reply Blocks ([katzenpostSURB] s).

This way in katzenpost there's also sender and receiver anonymity (unlinkability)

In loopix the user keys are also in the PKI, (are they also in katzenpost?)

2.4 Proposals 1, 2 and 1.1 arguments

See *Proposal 1 (P1) design*, *Proposal 2 (P2)* and *Proposal 1.1 (P11)*.

Pro and con arguments for the different proposals discussed after juga's presentation on January 12, 2021 [MixmailerSlides] .

2.4.1 Stratified topology

V. arguments against:

- attacks on the exits
- authorities decide the possition
- nodes should decide the route (free-routing)

j counterarguments:

- attacks on exits:
- do not deanonymize sender, nor their location, per se
- can be done by any operator that runs the exit:
  - reason why TLS is recommended (avoid MiTM)
- if target is the receiver, it is easy to find the "random" exit
- intelligence agencies try more sophisticated attacks trying to deanonymize the whole path
- it's the node that decide the possition in Loopix

2.4.2 Authorities

j arguments:

- they're needed so that:
  - all clients have the same view of the network, otherwise sybil attack
  - can reward/penalize nodes that go on and off, missbehave, etc.

V. counterarguments:

- it's the nodes which take those decissions.

2.4.3 TLS

j arguments:

- avoid clear metadata at last hop
- Let's Encrypt has helped a lot to do not depend on 3rd party entities

V. counterarguments:

- 0 trust on TLS

2.4.4 Message size

j arguments:

- can't pad while not possible to predict final size, what is only possible with same OpenPGP algo. and without compression
- need of fixed size (huge) padding so that the attacker doesn't know in which position of the route is the message

V. counterarguments:

- random padding, if message is too big and it's know it's at 1st possition in path, bad luck

2.4.5 OpenPGP

V.: pEp is not OpenPGP/MIME when 2 pEp clients talk, but it's OpenPGP/MIME compatible when the receiver is not pEp client.

2.4.6 Hidden headers

V.:

- agrees that the metadata is clear at last hop
- Outside (clear) *From* might not be the same as the inside (encrypted) *From*
- it has been implemented since years

2.4.7 GNS

V.: it's needed that GNUnet implements a GNS library

2.4.8 Other

V.: all this should be impemented in the engine including GNS resolving/registering

2.4.9 Katzenpost

j arguments:

- we can rewrite mail proxy. License of nodes running software doesn't matter cause pEp is not going to run them

V. contrarguments: not the technical solution we want

2.5 Threat model

See *Threat models* for an idea on threat modeling.

This is the threat model on *Proposal 1.1 (P11)*.

2.5.1 Arquitecture diagrams

See Nodes and messages, Deployment and Deployment with GNS

2.5.2 Threats on the software

2.5.3 Threats on the network transmission

See *Attacks* on common attacks to anonymity systems.

Adversaries

The adversaries here are all the adversaries mentioned in *Adversaries*.

Vulnerabilities

Attacks to which *Proposal 1.1 (P11)* is vulnerable:

- Replay Attacks
- Blending Attacks
- Passive subpoena attack
- Active subpoena attack
- Partition attack on client knowledge
- Tagging attack on headers
- Tagging attack on payload
- Attacks on multiple messages / large files: While messages are not splitted into packets, the packets don't have to end in the same node. But it's still vulnerable because of different sizes.
- Pseudospoofing
- Intersection attacks
- Timing and packet counting attacks: Even if padding is added, messages are still different sizes.

Mitigations

CHAPTER 3

Specification

Here specification. It is in markdown so that kramdown can convert it to xml for RFCs. It is recommended to read *Proposal1.1* and *Threat model*.

3.1 Introduction

TBD

3.1.1 Requirements Language

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 ([RFC2119] and [RFC8174]) when, and only when, they appear in all capitals, as shown here.

3.1.2 Terminology

TBD

Own vs existing.

Mix network/mixnet

Onion routing

Remailer

Node

Mix

client

pEp engine

GNUnet node

GNS Resolution

GNS Registration

GNS Delegation

Message

3.2 All the content

TBD

3.3 Security Considerations

TBD

CHAPTER 4

Roadmap

4.1 Roadmap

Overview of what is done so far:

- [X] Research on existing mix networks
- [X] Try existing mix networks and see how they could be adapted
- [X] Implement Proposal 1 including integration tests
- [X] Try GNUnet GNS and write tests
- [X] Document research, experiments, proposal

Next:

- [ ] Document threat model
- [ ] Write technical specification
- [ ] Implement the prototype, adapted to *Proposal 1.1 (P11)*, in the pEp engine

CHAPTER 5

References

5.1 References

On topics mentioned in this documentation.

5.1.1 Papers and RFCs

5.1.2 Web pages

5.1.3 Other

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Bibliography

[AnonTerms] http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[Batching] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.941.6881&rep=rep1&type=pdf,

[BibMixnets] https://bib.mixnetworks.org/

[Byzantine] https://people.csail.mit.edu/idish/ftp/Brahms-PODC.pdf

[DesignAnonymous] https://www.freehaven.net/anonbib/cache/wiangsripanawan-acsw07.pdf

[Gossip] http://pages.di.unipi.it/ricci/GossipBasedPeerSampling.pdf

[Loopix] https://arxiv.org/pdf/1703.00536.pdf

[MixCascadesP2P] https://www.esat.kuleuven.be/cosic/publications/article-523.pdf

[Mixmaster] https://tools.ietf.org/html/draft-sassaman-mixmaster-03

[MixminionPaper] https://www.mixminion.net/minion-design.pdf

[Sphinx] https://bib.mixnetworks.org/pdf/DBLP:conf/sp/DanezisG09.pdf

[Stopandgo] https://www.freehaven.net/anonbib/cache/stop-and-go.pdf

[SurveyRouting] https://dl.acm.org/doi/10.1145/3182658

[Trickle] https://www.freehaven.net/anonbib/cache/trickle02.pdf

[Trilemma] https://doi.org/10.1109/SP.2018.00011

[Uniform] www.irisa.fr/dionysos/pages_perso/sericola/PAPIERS/DSN13.pdf

[Untraceable] http://www.ovmj.org/GNUnet/papers/p84-chaum.pdf

[AnonymousRemailer] https://en.wikipedia.org/wiki/Anonymous_remailer

[ApplicationThreatModeling] https://owasp.org/www-community/Application_Threat_Modeling

[CERT] https://git.gnunet.org/gnunet.git/tree/src/gnsrecord/plugin_gnsrecord_dns.c#n130

[DHT] https://en.wikipedia.org/wiki/Distributed_hash_table

[gnunetGNS] https://en.wikipedia.org/wiki/GNUnet#GNU_Name_System

[gnunetGNUnet] https://en.wikipedia.org/wiki/GNUnet

[HuMixmaster] https://sarwiki.informatik.hu-berlin.de/Mixmaster_Remailer

[katzenpostSite] https://katzenpost.mixnetworks.org/

[katzenpostSURB] https://github.com/katzenpost/docs/blob/subscription_plugin.0/drafts/mix_plugin_subscription.rst

[Mailproxy] https://github.com/katzenpost/mailproxy

[MAPI] https://en.wikipedia.org/wiki/MAPI

[Mixmailer] https://gitea.pep.foundation/pEp.foundation/mixmailer

[MixmailerSlides] https://gitea.pep.foundation/pEp.foundation/mixmailer_slides

[Mixminion] https://en.wikipedia.org/wiki/Mixminion

[MixNetworks] https://en.wikipedia.org/wiki/Mix_network

[MixOnion] https://ritter.vg/blog-cryptodotis-mix_and_onion_networks.html

[NymServer] https://en.wikipedia.org/wiki/Pseudonymous_remailer#Contemporary_nym_servers

[Nymtech] https://nymtech.net/

[OnionRouting] https://en.wikipedia.org/wiki/Onion_routing

[PET] https://en.wikipedia.org/wiki/Privacy-enhancing_technologies

[RemailerNym] https://en.wikipedia.org/wiki/Pseudonymous_remailer

[SPF] https://en.wikipedia.org/wiki/Sender_Policy_Framework

[STRIDE] https://en.wikipedia.org/wiki/STRIDE

[ThreatModelingOutputs] https://owasp.org/www-community/Threat_Modeling_Outputs

[TypeI] https://en.wikipedia.org/wiki/Anonymous_remailer#Cypherpunk_remailers,_also_called_Type_I

[TypeII] https://en.wikipedia.org/w/index.php?title=Anonymous_remailer&section=4#Mixmaster_remailers,_also_called_Type_II

[TypeIII] https://en.wikipedia.org/wiki/Anonymous_remailer#Mixminion_remailers,_also_called_Type_III

[AttacksAnonymitySystems] https://www.blackhat.com/presentations/bh-usa-03/bh-us-03-sassaman-dingledine/bh-us-03-dingledine.pdf, https://www.blackhat.com/presentations/bh-usa-03/bh-us-03-sassaman-dingledine/bh-us-03-sassaman.pdf, https://www.youtube.com/watch?v=RQ1ikYB_1LY, https://www.youtube.com/watch?v=tAuCX9V034Q