Attacks on Anonymity Systems:
The Theory

Roger Dingledine
http://freehaven.net/

Len Sassaman
http://anonymizer.com/

One talk, two slots

Attacks on Anonymity Systems: The Theory

Attacks on Anonymity Systems: The Practice

We focus on high-latency systems (remailers)

Part one: outline

Adversaries and threat models

Walkthrough of the Mixminion design process

Design choices, economic issues

Many people need anonymity

Individuals are tracked and profiled daily

- Imagine your dossier in twenty years
- (If that doesn't scare you, think of your kids)

Political dissidents in oppressive countries

Governments want to do operations secretly

Corporations vulnerable to traffic analysis:

- VPNs, encryption don't block corporate espionage

Hide users with users

Anonymity systems use messages to hide messages (the more noise, the more anonymous something in that noise is)

Senders are consumers of anonymity, and providers of the cover traffic that creates anonymity for others

Users might be better off on crowded systems, even if those systems have weaker anonymity

designs

Strong anonymity requires distributed trust

An anonymity system can’t be just for one entity

(even a large corporation or government)

You must carry traffic for others to protect yourself

But those others don’t want to trust their traffic to just one entity either

Adversary characteristics

External (wires) or Internal (participants)

Passive or Active

Local or Global

Static or Adaptive

Some sample adversaries

Global passive adversary: watches all links

Rogue operator: runs one or a handful of nodes

External attacker: can inject/modify some traffic

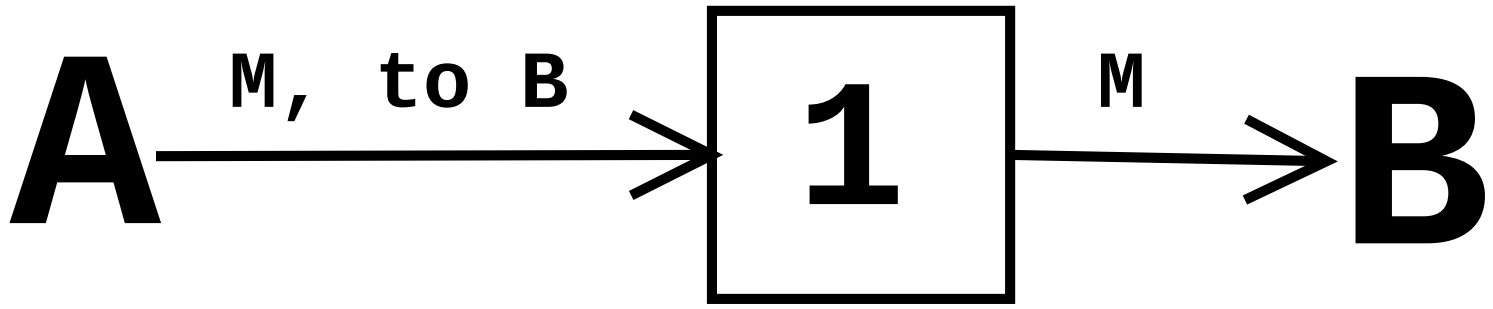
Mixminion threat model

All three:

- Global passive adversary -- can observe *everything*
- Owns some of the nodes
- Can inject, modify, delete some traffic

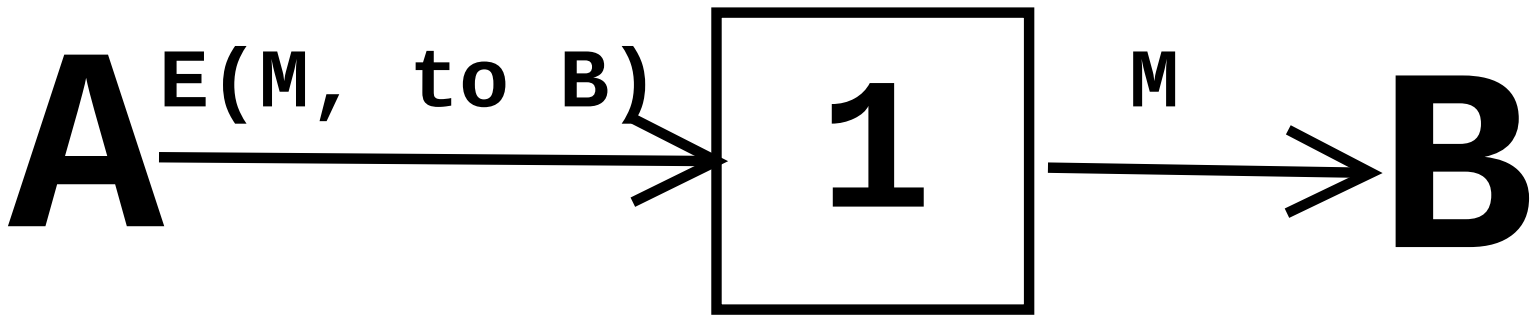
We are not real-time, packet-based, or steganographic

Direct Forwarder



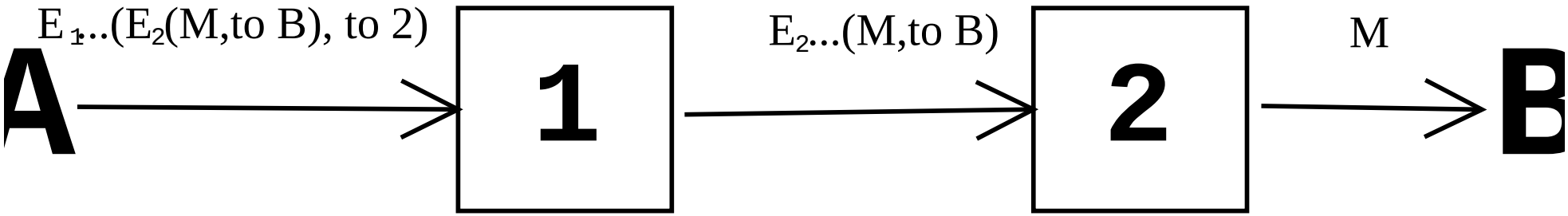
But: an observer of Alice can just read M and know it’s going to Bob

Add Encryption



But: 1 still knows Alice sent M to Bob

Multiple hops



Assume not all hops will collude and reveal A

But: How do you know what the servers are?

Statistics servers

(aka directory servers)

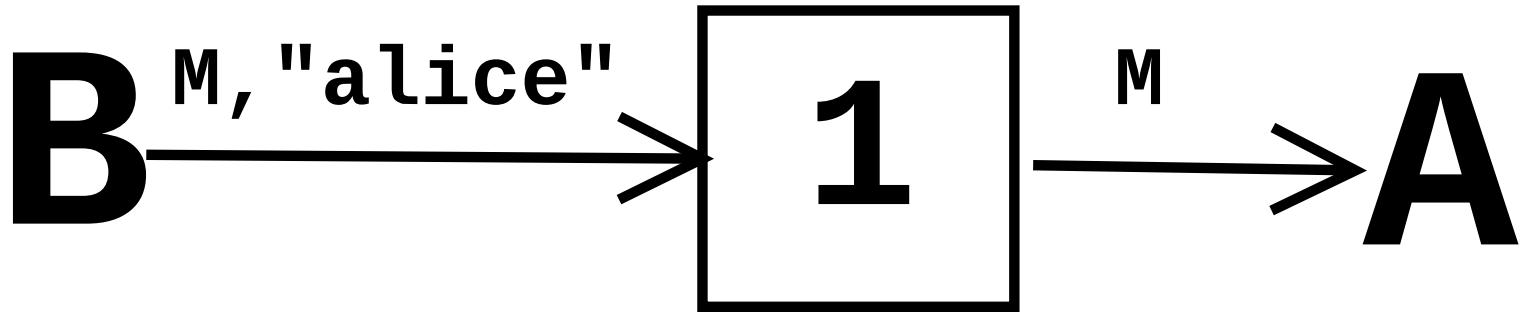
```
Mixmaster    Latent-Hist   Latent  Uptime-Hist   Uptime  Options
-------------------------------------------------------------------
winter       1110320100l0    :42   ++++++++++++  100.0%   PR  O
xganon        000000000000   :03   ++++++++++++  100.0%   PR
green         00000000000?   :09   ++++++++++0   97.8%    2  O
lcs          151231221221   1:30   +++++++++7++   97.8%    M
```

Have several servers to avoid single point of failure

They can send test messages and tell users which nodes are up

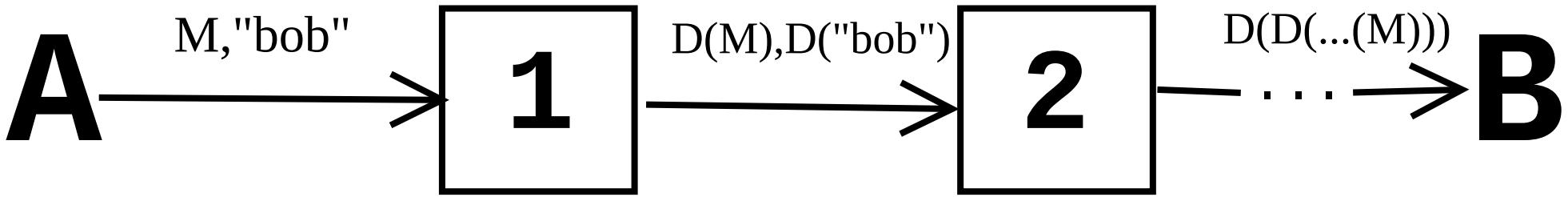
Direct Reply

(Trying to hide A's location)



"alice"=an4691@anon.penet.fi
(A has told 1 her location.)

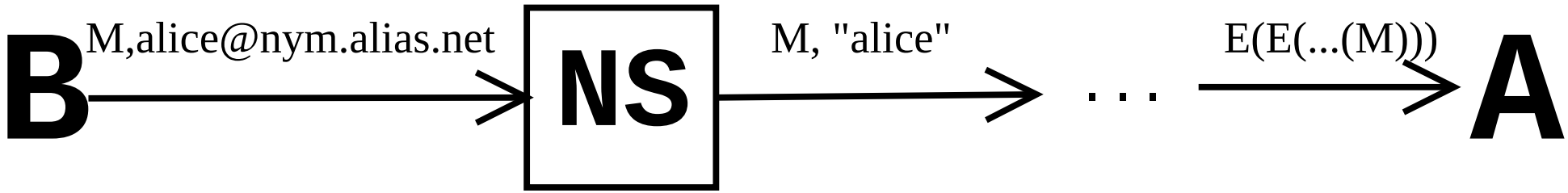
Reply Blocks



"alice" = 1, E_1(2, ...E_n(A))

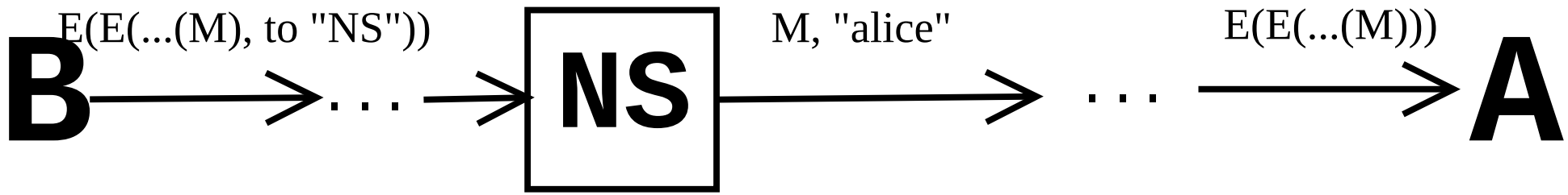
Hard for B to get a reply block from A.

Nymserver



NS knows A’s reply block but not her location.

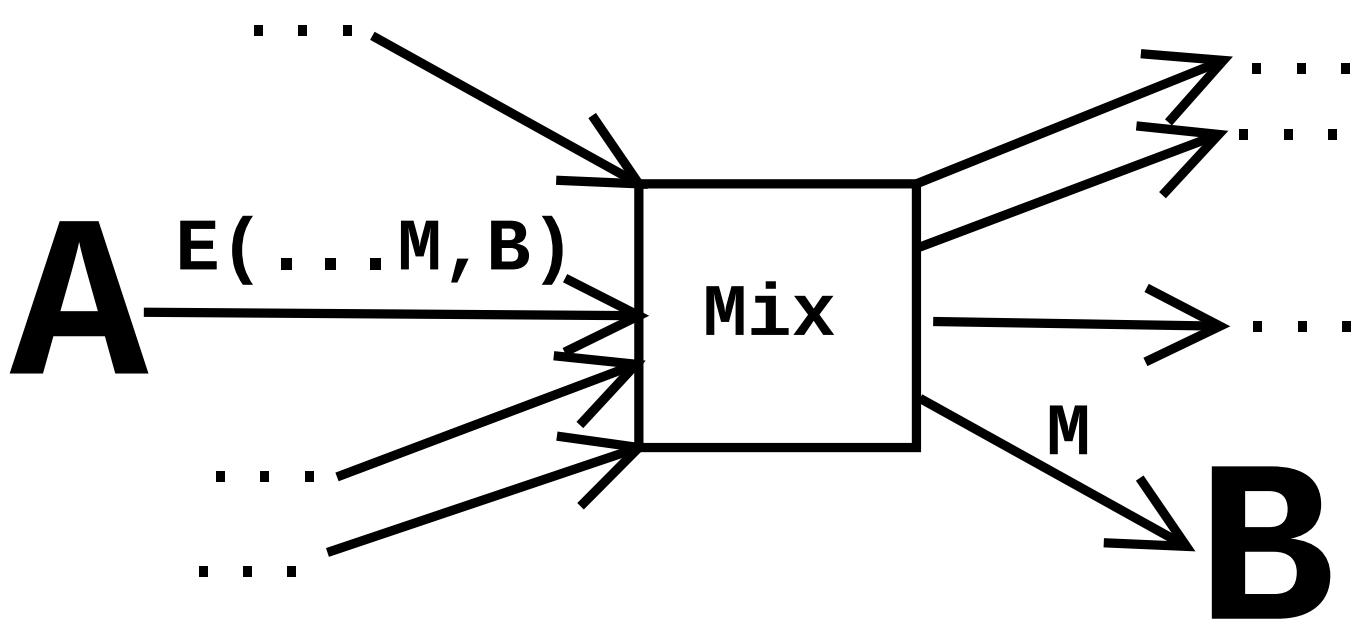
Anonymized Reply



NS doesn’t know A or B

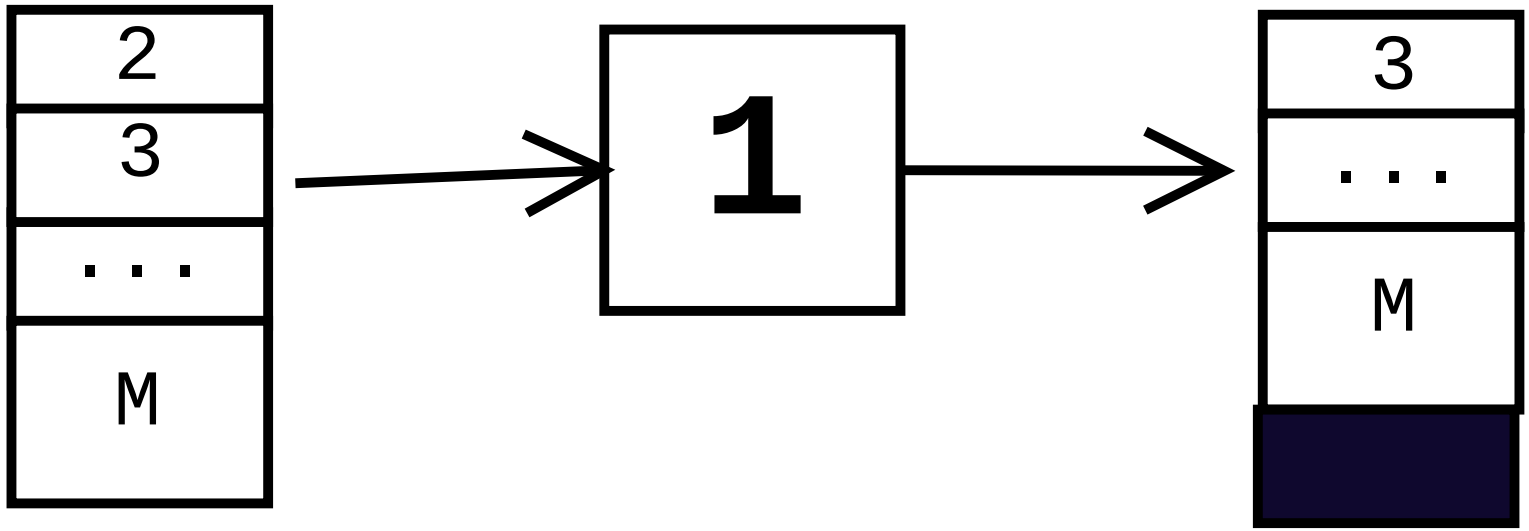
If you stop here you get type 1 (cypherpunk) remailers.

Batching and Mixing



But: Different-sized messages can still be distinguished.

Fixed length messages: repadding



- Add random junk to the bottom to replace the header you strip off
- Everything's encrypted, so it looks ok.

But: Replay attacks -- a given message always

Replay cache

When a message comes in, hash it and add it to the replay cache.

If it’s already in the cache, drop it.

But: you have to remember all the hashes forever!

Expiration dates

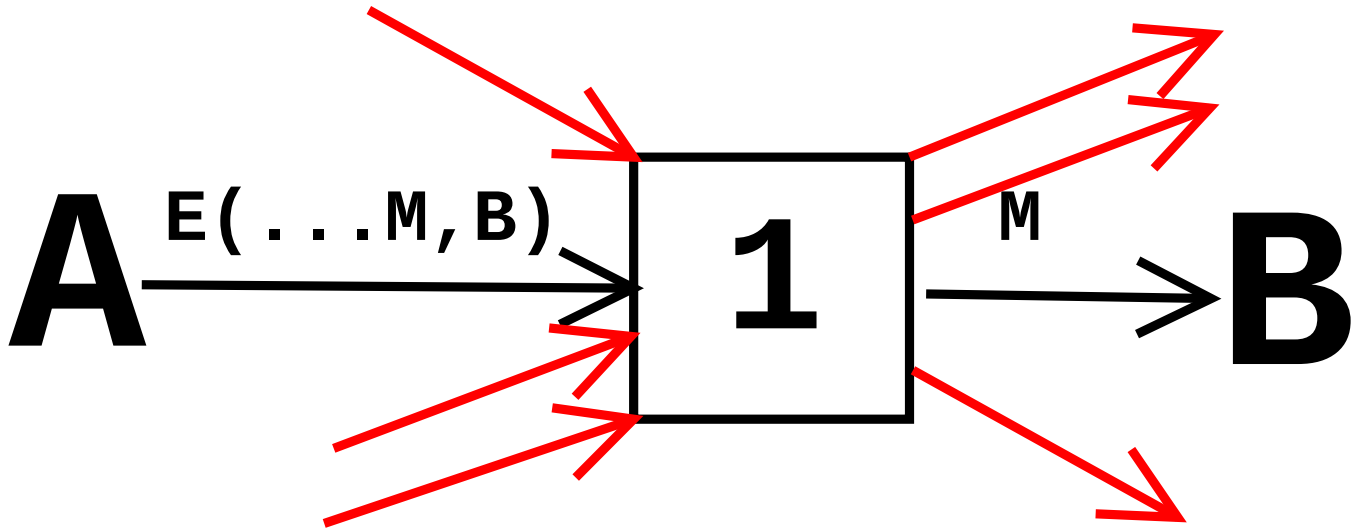
Exp date is chosen randomly between 3 days ago and 3 days from now.

Each node checks exp date; if more than 7 days old, drop.

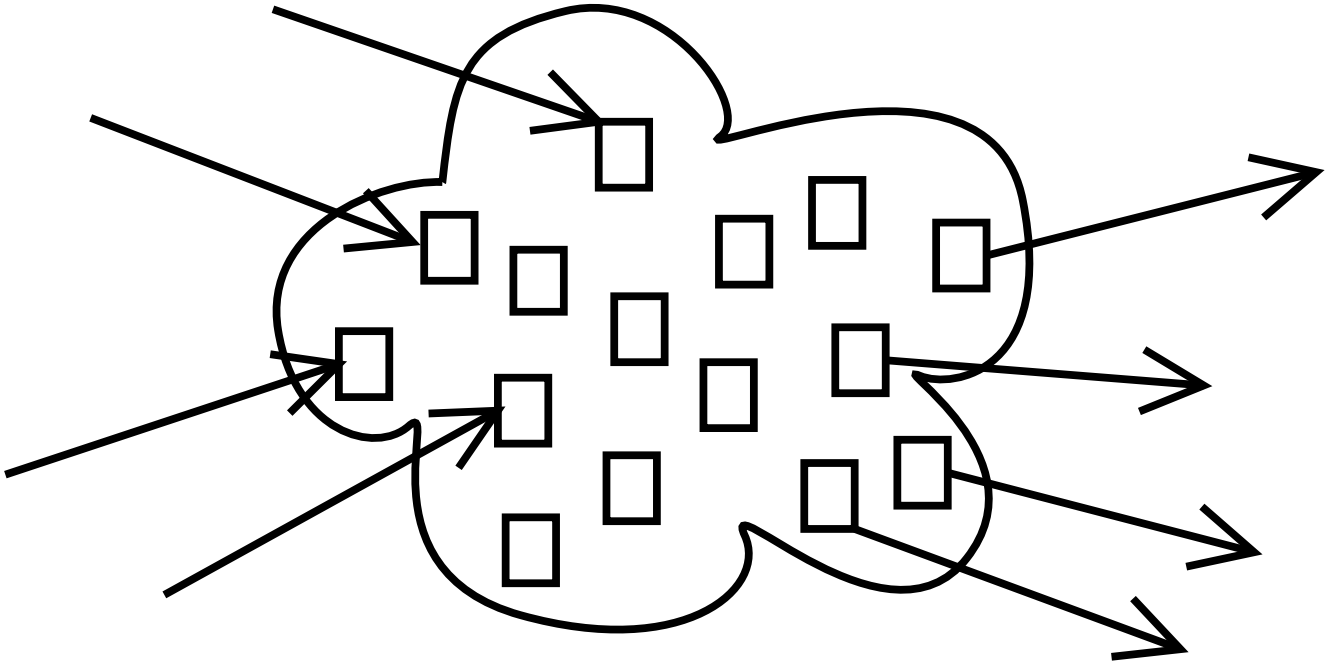
Now adversary can't tell when the message was sent from its exp date; and servers can forget hashes that are >7 days old.

Flooding attack

But you can flood a node so you know all but one message in the batch.



Pooling



Not all messages come out at each flush.
Keep a minimum number in the pool, always.

Now it’s harder to target an individual

But: Trickle attack -- what if you’re the only one who sends a message into the node in a given interval?

More broadly, what if you’re the only one who sends a message into the whole network, in that interval?

Dummy messages

Users sometimes send decoy messages even if they have nothing to send.

Hopefully there will be enough messages that the adversary will be confused.

Dummies go several hops and stop (hard to decide convincing destinations).

If you stop here, you get type 2 (Mixmaster) remailers.

Passive subpoena attack

Eve can record messages for later subpoena
She can also recognize her own messages, which helps with flooding attacks

Fix: Link encryption with ephemeral keys (rekeyed every message / few minutes)

Active subpoena attack

Mallory can still record messages from the node she runs, and arrive later with a subpoena.

Fix: Periodic key rotation

Partition attack on client knowledge (1)

Adversary can distinguish between clients that use static node lists and clients that frequently update from the directory servers.

Fix: Clients must all use the same algorithm for updating from the directory servers.
Directory servers must be part of the spec!

Partition attack on client knowledge (2)

Directory servers can be out of sync; evil directory servers can give out rigged subsets to trace clients.

Fix: DSs must successively sign directory bundles; a threshold of servers is assumed good.

What do users do if the DSs can’t agree?

Partition attack on message expiration date

Delaying a message a few days will push its exp date to one end of the valid window -- so they won’t be uniformly distributed.

Fix: No expiration dates. Keep all hashes until key rotates.

Tagging attack on headers

Mixmaster headers have a hash to integrity-check the fields for that hop. But it doesn’t check the rest of the header.

So we can flip some bits later in the header, and if we own the node later in the path that corresponds to the header we just broke, we can recognize the message.

We must make the hash cover the entire header.

Tagging attack on payload

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

Fix: Make the hash cover the payload too.

Still using Cypherpunk replies

No replay detection, no batching, messages change length at each hop, etc.

Fix: Do all this stuff for replies too.

Since we want to *encrypt* replies at each hop, use a cryptosystem where decrypt is as strong as
encrypt.

But you can’t write a reply block without knowing the payload!

Since the author of the reply block can’t guess the right hashes for the payload, we’ve reintroduced the payload tagging attack.

Actually, that’s ok. Since we’re *encrypting* at each hop, only the recipient can recognize the tag.

But forward messages and replies must now be distinguishable

Forward messages need hashes, and replies can’t have them.

Assuming replies are rare relative to forwards, replies are easy to track.

We support three delivery types

Forward messages, only Alice is anonymous

Direct replies, only Bob is anonymous

Anonymized reply messages where Alice *and* Bob are anonymous

(Parties that get anonymity must run our software.)

Messages have two headers and a payload

Divide the path into two legs, one for each header

- For forward messages, Alice chooses both legs
- For direct replies, Alice can use the reply block directly
- For anonymized replies, Alice chooses the first leg and uses Bob’s reply block for the second.

Legs are connected by the Crossover Point

One of the hops in the first header is marked as a *crossover point*

At the crossover point, we decrypt the second header with a hash of the payload, and then swap the headers.

Forward messages are anonymous:

If the second header or the payload are tagged in the first leg, then the second header is

unrecoverable.

If tagged in the second leg, we've already gotten anonymity from the first.

Replies are anonymous:

The adversary can never recognize his tag.

Multiple-message tagging attacks

If Alice sends multiple messages along the same path, Mallory can tag some, recognize the pattern at the crossover point, and follow the rest.

Only works if Mallory owns the crossover point.

Fix: Alice picks k crossover points
(and hopes Mallory doesn’t own most of them)

Nymservers out of single-use reply blocks

Work like pop/imap servers

User anonymously sends a bunch of reply blocks to receive the mail that's waiting for him.

If you stop here, you get the current Mixminion remailer design.

Open problem: reputation on the directory servers

How do we let clients learn which nodes are good, without:

Letting the adversary do partitioning attacks on clients

Letting the adversary get more traffic by behaving well

Open problem: trickle attack on directory servers

Malicious nodes can hold a message and release it later, when circumstances are different.

More broadly, we’re still in an arms race against flooding and trickle attacks

Open problem: long-term intersection attack

The fact that not all users are sending messages all the time leaks information.

By observing these patterns over time, we can learn more and more confidently who is sending mail, to whom, when, etc.

Major unsolved problem in anonymity systems.

Topology options

Cascades, free-route, restricted-route

Synchronous vs asynchronous

Recipient anonymity

Reply onions / reply blocks

Nymservers

Rendezvous points

Unobservability

Running your own node -- plausible deniability

Cover traffic

An Economics of Anonymity

- Unlike encryption, it's not enough for just one person to want anonymity: the infrastructure must participate
- Systems need cover traffic (many low-sensitivity users) to attract the high-sensitivity users
- Most users do not value anonymity much
- Weak security (fast system) can mean more users
  - which can mean *stronger* anonymity
- High-sensitivity agents have incentive to run nodes
  - so they can be certain first node in their path is good
  - to attract cover traffic for their messages
- There can be an optimal level of free-riding

Up next...

Which of these attacks do we see in practice?
How much damage do they do?