Single-hop proxies

- Most popular, easiest to deploy
- Single point of failure (legal, technical)
- Anonymizer, Safeweb, ...

A MIX node

- Messages change appearance after decryption
- Each MIX batches and reorders messages
- Messages are all the same length
- Store and forward (slow) to maintain anonymity sets

Free-route MIX networks

- User picks a path through the network
- Goal is to hide message's *path*
- Needs dummy traffic (inefficient, poorly understood) to protect against global adversaries (lots of traffic may work too?)
- Example: Mixmaster, Mixminion

A MIX cascade

- Use multiple nodes to distribute trust: any one node can provide anonymity.
- Anonymity comes from the more *users*, not more nodes.
- Assumes a global adversary
- Dangers: trickle attacks, easy to watch endpoints
- Example: Web MIXes, Java Anon Proxy

Crowds

- Plausible deniability for web browsing
- Users forward requests within their crowd
- At each forward, with prob p the request is forwarded to another member, else it goes to the webserver.
- webserver doesn't know who made the request.
- No encryption/mixing: totally vulnerable to global adversary

Zero Knowledge's Freedom Network

- Connection-oriented (low latency)
- Paid ISPs to run Freedom nodes
- Tunnelled all traffic (udp, tcp, icmp — everything) through the Freedom network
- But not enough users to make it viable

Onion Routing

- Connection-oriented (low latency)
- Long-term connections between Onion Routers
  Link padding between the routers
- Aims for security against traffic analysis, not traffic confirmation
- Users should run node, or anonymize connection to first node, for best privacy

(Onion routing intro)

Some technical problems for Onion Routing:

Convenient/Usable Proxies

- We can use anything that has SOCKS support. But we must strip identifying data: new proxies?
- Another approach is to intercept all traffic – otherwise we need to modify applications so they don't leak info.
- ...and nobody will use it if we need all these proxies (not true: p2p systems?)

Ideal threat model

- Global passive adversary – can observe *everything*
- Owns many of the nodes

Link padding and topology

- Remember that our goal is to hide the *path*
- Without link padding, adversary can observe when new connections start, and where they go.
- n^2 link padding is insane, but anything less seems unsafe.
- Open problem: what's the right compromise?

Timing attacks

- If the adversary owns two nodes on your path, he can recognize that they're on the same path
- Works passively (counting and watching packets and timing) or actively (delaying and batching packets so they're optimally recognizable).
- An external active adversary can do this by saturating links or otherwise delaying messages into a certain profile which is recognizable downstream.

Tagging attacks

- Onion routing uses a stream cipher to encrypt the data stream going in each direction.
- An adversary owning a node – or a link! – can flip a byte in the data stream and look for an anomalous byte at the exit point (say, when it talks to a webserver).
- This sort of thing is generally solved by including a hash, but it's more complex than that.

Long-term intersection attacks

- 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.

More realistic threat model

- We must retreat to protecting against *traffic analysis*, not *traffic confirmation*.
- Reasonable threat model still an open problem too.

Oh yeah, and I wrote some Onion Routing code

- It's GPLed, but the Navy is sitting on it. Stay tuned.

(Short break)

Next: *Anonymity is hard for economic/social reasons too*

abuse

Anonymity is hard for economic/social reasons too

- Anonymity requires *inefficiencies* in computation, bandwidth, storage
- Unlike encryption, it's not enough for just one person to want anonymity — the infrastructure must participate

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

More users is good

- High traffic $\Rightarrow$ better performance
- Better performance $\Rightarrow$ high traffic
- Attracts more users: faster *and* more anonymous

But trust bottlenecks can break everything

- Nodes with more traffic must be more trusted
- Adversary who wants more traffic should provide good service
- (and knock down other good providers)
- Performance and efficiency metrics *cannot* distinguish bad guys from good guys

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

Can we fund it by offering service for money?

- Freedom taught us that end-users won't pay enough for strong anonymity
- (Ok, ok, it's more complicated than that.)

Can we get volunteers to run nodes?

- Liability, especially for exit nodes
- Having lots of nodes might work, but making an example of a few well-chosen nodes can scare everybody
- We can allow nodes to set individual exit policies
- Remains an open problem

Pseudospoofing: volunteers are a danger too

- Are half your nodes run by a single bad guy?
- Global PKI to ensure unique identities? No.
- Decentralized trust flow algorithms? Not yet.
- Still a major open problem for dynamic decentralized anonymity systems

Need to manage incentives

- Users have incentive to run a node, to get more anonymity. That's a good start.
- Dummy traffic can help maintain anonymity – but why should others send dummy traffic to help *your* anonymity?
- If anonymity for all requires each user doing similar things, how do we deal with users who don't want as much anonymity?

Customization and preferential service are risky (1)

- It's tempting to let users choose security and robustness parameters
- Eg, how many replicas of my file should I create? or how many pieces should I break my file into?
- But a file replicated many times stands out.

Customization and preferential service are risky (2)

- We'd like to let clients customize to barter better, e.g. in systems like Mojonation
- We'd like to let users pay (or pay more) for better service or preferential treatment
- But the hordes in the coach seats are better off anonymity-wise than those in first class.

Conclusion 1: we're screwed

- Usability is a *security* objective: anonymity systems are nothing without users.
- It's critical that we integrate privacy into the systems we use to interact.
- But it's hard enough to build a killer app.
  It's going to be really really hard to solve all the factors at once.

Conclusion 2: more research remains

- Our current directions aren't going to work, from an incentive and usability perspective. We need to rethink.

Synchronous systems

- Each message has a deadline by which the node must pass it on
- Length of paths is fixed, paths might even be public
- Anonymity is now based on size of batch at widest point, even for free-route systems
- Improves flooding/trickle attacks
- But harder to synchronize, especially for low-latency systems

Privacy Enhancing Technologies workshop

March 26-28, 2003

Dresden, Germany

http://petworkshop.org/