Why is anonymity so hard?

Roger Dingledine

The Free Haven Project

Talk overview

- Motivation
- Anonymity systems
- Why anonymity is hard
- Example: Onion Routing

Many people need anonymity

- Political dissidents in oppressive countries
- Governments want to do operations secretly.
- Corporations are vulnerable to traffic analysis (corporate espionage) — VPNs, encryption don't cut it.
- Individuals are tracked and profiled daily. Imagine what they'll have in your dossier in twenty years.
- (If that doesn't scare you, think of your kids.)

Single-hop proxies

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

Adversary characteristics

- External (wires) or Internal (participants)
- Passive or Active
- Local or Global
- Static or Adaptive

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

A MIX cascade

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

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
- Example: Mixmaster

Crowds: anonymous web browsing

- "Blending into a crowd"
- Users forward requests within their crowd
- At each forward, with probability p the request is forwarded to another member; else it goes to the webserver.
- So the webserver cannot know which member of the crowd made the request.
- No encryption/mixing: totally vulnerable to global adversary

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

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

But anonymity is hard

- Anonymity requires *inefficiencies* in computation, bandwidth, storage

- Unlike encryption, it's not enough for just one person to want anonymity — the infrastructure must participate

Other people provide your anonymity (noise)

- The more noise, the more anonymous something in that noise is

- You're always better off going where the noise is

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

Strong anonymity requires distributed trust

- An anonymity system can't be just for one entity
- (even a large corporation or government)
- So 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

Even customization and preferential service are risky

- It's tempting to let users choose security 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.

Even customization and preferential service are risky

- 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.

An example: Directory servers

- Distribute location, capabilities, key info, performance stats
- A single directory server is a point of failure
- Redundant directory servers: must be well-synchronized to avoid partitioning attacks
- Can distinguish between clients that use static lists and clients that update frequently

Directory servers (cont)

- But even if client information is uniform, nodes can still do trickle attack: hold message until other clients have different information.

- Introducing reputation means adversary has new avenue to manipulate client information

- Tension between giving clients accurate timely information, and preventing adversary from manipulating client behavior

Conclusion: 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.

- Our current directions aren't going to work. We need to rethink.