

How to remain anonymous on the Internet?

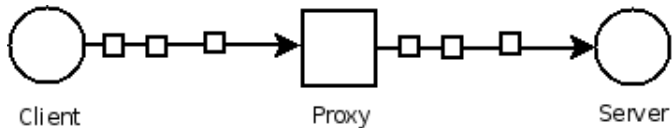
There are valid reasons to remain anonymous on the internet

- ▶ Prevention of linking attacks
- ▶ Suppressive regimes
- ▶ Control of personal information, medical for example.
- ▶ Law enforcement use it
- ▶ However, criminals of course use it

But every connection has an IP address source and destinations. How to remain anonymous?

The basic approach

Use a HTTP/SOCKS proxy



Using a Proxy

There are problems with the basic approach of using a proxy

- ▶ If the attacker can observe traffic entering and leaving the network, it's no problem to see a plaintext request enter and the same request leave.
- ▶ We know how to deal with that though. Make sure the client uses an encrypted connection when connecting to the proxy. Plaintext correlation is no longer possible.

Using a Proxy

Is that enough ?

- ▶ The incoming data is encrypted, but information is still leaking.
- ▶ Even though it is encrypted, the size of the data remains the same, or at least very similar. An observer can measure the amount of data coming in one connection and check what's coming out.
- ▶ To prevent this, we introduce dummy data. Insert padding into the incoming data, so preventing size comparisons.

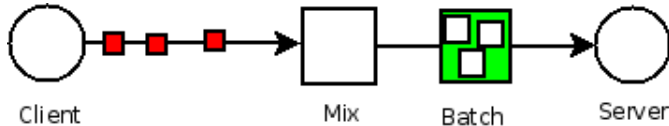
Using a Proxy

Still not enough

- ▶ We still have a problem. Timing information.
- ▶ The proxy doesn't take long to handle a request and send it on.
- ▶ Observe the time that requests come in at and when they leave. Straightforward to compare and determine which incoming is which outgoing.
- ▶ Introduce delay. Delay incoming streams for a random amount of time.

A basic MIX

These elements are what a basic MIX, described by David Chaum, consists of.



Basic MIX

- ▶ Designed for e-mail
- ▶ Incoming messages are padded to a specific size.
- ▶ They are encrypted with the public key of the MIX server
- ▶ When messages arrive they are put into a delay queue
- ▶ Outgoing messages have their sender name/e-mail replaced with the MIX's.
- ▶ When sending messages, send them in mixed up batches.

This is pretty good. But there is still a large flaw.

A single point of failure

We are relying on a single machine to preserve anonymity. If the machine itself is compromised, so is anonymity. We need to mitigate the risk somehow

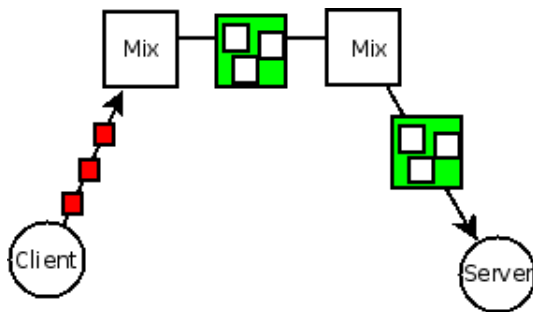
- ▶ Solution: Add more mixes

Cascade Mix

CA645

Gavin O' Gorman

Anonymous
Networks

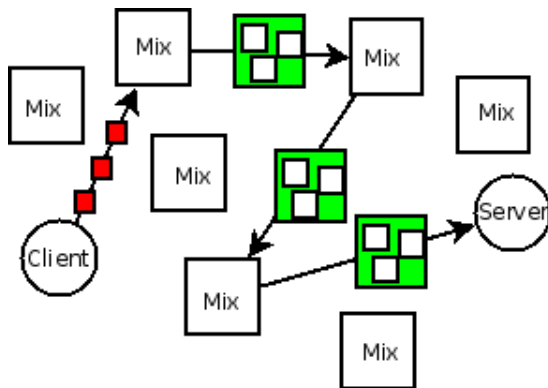


Free Cascade Mix

CA645

Gavin O' Gorman

Anonymous
Networks



Low latency

This sort of mix and batching works well for high latency traffic, such as e-mail. A large delay can be tolerated. But what about low-latency, interactive traffic such as HTTP, VoIP etc. Several low latency anonymous networks have been proposed

- ▶ Onion Routing
- ▶ Crowds
- ▶ Piplinet/Freedom
- ▶ Tarzen

- ▶ Developed in association with the US Naval Research laboratory
- ▶ Data packets are wrapped up in encryption and sent through the mix network - the layers of encryption are the onion
- ▶ Tor is the latest implementation. More on this.

Crowds

The Crowds maxim is 'anonymity prefers company'.

Designed for anonymous web browsing, Crowds is very loosely based on the mix model but with a number of substantial differences.

- ▶ Crowds focuses purely on web browsing, that is, HTTP
- ▶ A group of disparate computing nodes form a crowd. These client nodes are referred to as jondos (John Doe, an anonymous person).
- ▶ A jondo wishing to connect to a web server anonymously establishes a connection via several other jondos in the crowd and sends his/her HTTP request via this connection. The response is returned along this same route. Connections are established by flipping a weighted coin and deciding based on this whether to connect to another node, or use the current node as the exit node.
- ▶ A single encryption key is used

PipeNet/Freedom

Freedom and Pipenet are described together here as they are essentially the same design as onion routing. They both consist of a set of core nodes which reroute data, data which is encrypted in an onion like manner. There are some differences

- ▶ Pipenet utilizes a constant bi-directional stream of traffic between consecutive machines on a route in order to prevent traffic analysis. This design is not practical as it utilizes too much bandwidth.
- ▶ Freedom has the option to choose packet size and implements an application level system of pseudonyms and email addresses to aid in the users anonymity. This level is irrelevant to the lower level TCP anonymity. Freedom was a commercial enterprise, since shut down.

Based on the peer to peer paradigm, Tarzan is designed to overcome the possibility of a set of core router nodes being compromised.

- ▶ Given the potentially large amount of peer to peer clients, all of which can route data, there is a lower probability of all nodes on a path being compromised.
- ▶ Tarzan uses standard P2P techniques for looking up further peers, such as Distributed Hash Tables etc.
- ▶ Data is encrypted, again using the onion format, before being sent onto the next node in the route.

Tor is the most popular anonymous network. How does it work?

- ▶ Uses a free cascade architecture, with telescoping circuits.
- ▶ It's an implementation of onion routing, so the same technique for encrypting data
- ▶ Used a fixed cell size of 512 bytes
- ▶ Directory servers used to update clients on onion routers
- ▶ Offers Hidden Services

- ▶ The Tor client runs as a SOCKS proxy. Applications with SOCKS support simply use it as a proxy.
- ▶ When one wishes to connect to a server, Tor establishes a circuit
- ▶ A list of routers are chosen and an appropriate exit router.
- ▶ A connection is sent to the first router. Keys are negotiated using Diffie-Hellman
- ▶ An extension request is sent to the first router. It connects to a second router and shuttles data back and forth. Again, DH is used to establish a key. This continues onto the exit router, which establishes a TCP connection to the target server. Data can be sent across the complete circuit at this point.

Hidden Services

Tor is typically used to allow for sender anonymity, but can also be used for receiver and sender.

- ▶ A user is running a web server and wants people to be able to connect, but the user wishes to remain anonymous
- ▶ A simple explanation is that the user tells an onion router to advertise this web server service.
- ▶ Alice, the client, connects to the advertising onion router.
- ▶ The advertising onion router then routes the connection to the server.
- ▶ Because it's running through the Tor network, it can bypass NAT and firewalls

There are a number of practical attacks, theoretical attacks and general issues that one needs to be aware of when using Tor

Information leaking will compromise anonymity

- ▶ Javascript/Java applets/Flash can make network connections independently of the browser. They may not be routed through the proxy and so can purposefully, or accidentally expose an IP.
- ▶ DNS requests must be made through the Tor network, otherwise an observer can see requests on the way out, or at the DNS server
- ▶ Data leaving the exit node is not encrypted if using plain HTTP !

Exit Node compromises

It is relatively common for exit nodes to be setup, purely to examine outgoing traffic. Some famous incidents

- ▶ in 2007, Dan Egerstad ran 5 Tor exit Nodes and examined traffic transiting the node.
 - ▶ He published 1000 e-mail usernames and passwords.
 - ▶ Embassies and Diplomats e-mails were observed. Iran, India, Australia, Japan etc
- ▶ Blackhat conference two weeks ago - a SSL MITM attack
 - ▶ Pick up connections going to port 443 and re-direct them to 80.
 - ▶ Run a SSL proxy in the middle with a valid signed cert and the https:// appears as does a padlock icon. No error messages
 - ▶ Some tricks with unicode to spoof the url

Remember. Tor offers Anonymity. NOT Confidentiality.

Theoretical attacks

- ▶ Website Fingerprinting
 - ▶ Connect to a large number of websites through Tor
 - ▶ Record the number of packets, packet timing etc
 - ▶ Build a database of this and then compare to traffic seen on a real Tor network.
- ▶ Brute force clogging attack. Global adversary
 - ▶ Determining what route a connection is taking.
 - ▶ Observe traffic at the server
 - ▶ Blast traffic at potential routers and see if there is an interruption in the traffic on the server.

A brief look at even more theoretical work

How can we measure the anonymity that Tor offers ? How can a user know the probability that they will be identified?

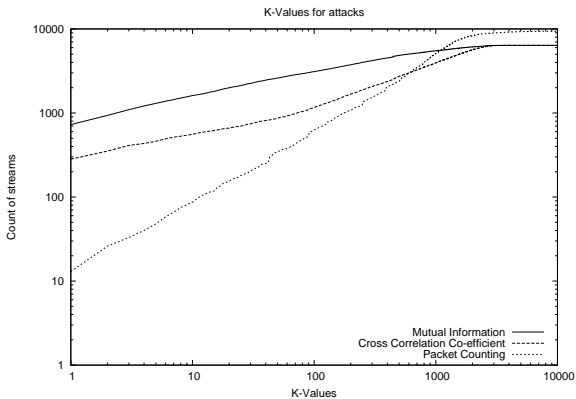
- ▶ One way of measuring the definite level of anonymity is to implement correlation attacks on traffic entering and exiting the Tor network
- ▶ Find out how easy or hard it is to correctly identify which stream coming into the network, is which stream leaving the network
- ▶ This will give a metric for anonymity
- ▶ It's not possible to do however with the deployed Tor network

One solution - Simulation

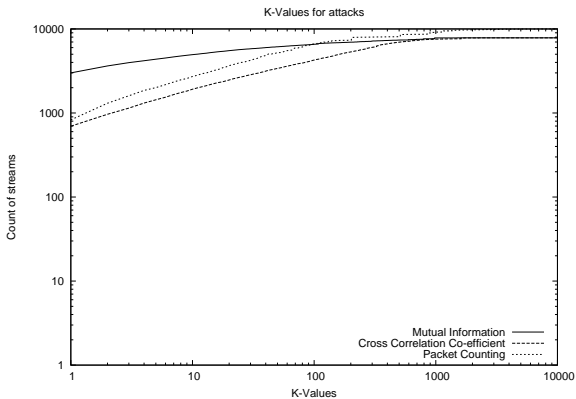
There are software packages that allow one to build a simulation of a network. They simulate Ethernet, TCP/IP, UDP, HTTP etc.

- ▶ So, take one of these simulation packages and recreate Tor as a simulation.
- ▶ Setup a topology with HTTP client, server, tcp/ip routers, onion routers and exit routers
- ▶ Set off the simulation and record all the traffic entering and exiting the network
- ▶ Split the traffic into streams and then use correlation algorithms to compare the streams

Results for real Tor network

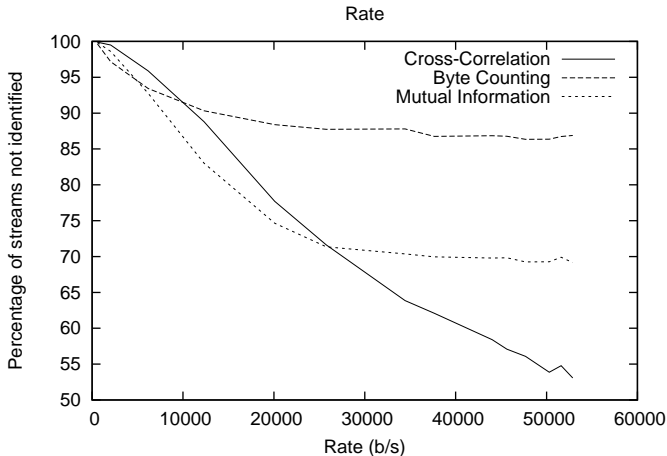


Results for simulation



Performance

Now we are in a position to start measuring other elements, such as traffic rate, or performance versus anonymity



That's it

CA645

Gavin O' Gorman

Anonymous
Networks

That's it from me, hope you enjoyed it. If you come up with projects and are looking for a supervisor, I may be interested!