

Onion Routing and Tor

Tonnerre Lombard

September 1st, 2005

Abstract

This document looks deep into the Tor protocol and its origins, in order to explain its functionality. It also takes a look at the internals of Tor and points out problems and questions to be asked in the future.

This document is also supposed to be a big help to everyone who wants to understand and operate Tor clients and/or servers.

Contents

1	Who needs anonymity?	2
1.1	Wouldn't criminals like to be anonymous, too?	2
1.2	What about intrusion?	3
2	The problem	3
3	Requirements to a solution	4
4	Prior approaches	4
4.1	Proxies	4
4.2	Mixes	5
4.3	First generation Onion Routing	6
5	The principle of Onion Routing	6
5.1	History	6
5.2	Mode of operation	7
5.3	Beware of the DNS!	8
5.4	So who are these nodes?	8
5.5	Hidden services	8
6	Problems	9
7	Design questions	10
8	Facts and numbers	10

9	Installing Tor	10
9.1	Privoxy for web browsing	11
9.2	Socat for SSH	12
10	Web links	13

1 Who needs anonymity?

- Journalists, dissidents, whistleblowers (Indymedia, bloggers, Iran, Tibet)
- Censorship resistant publishers and readers (e.g. investigations on radical right-wing organizations without being arrested as a Nazi)
- Socially sensitive communication (e.g. physicists, talking groups, forums about abuse etc.)
- Law enforcement (anonymous hints or crime reports)
- Companies (protection against spies, hiding information about suppliers and customers, etc.)
- Governments (anonymous polls/elections, exchange between untrusted governments)
- You (Who do you mail with? What websites do you visit? Where do you work? (→ stalking) What do you buy? What physicists do you visit? (→ abortion?) What kind of books do you read? (→ Amazon wishlist case))

1.1 Wouldn't criminals like to be anonymous, too?

That would indeed be a logical assumption. However, criminals mostly couldn't care less about the safety of other people around them, so they already have a whole lot of other possibilities to be anonymous:

- Identity theft
- Being unidentified for a long-enough period of time (e.g. suicide bombers, martyrion in jail)
- Motivation to use far more complex anonymity systems
- Motivation to *buy* anonymity

Neither normal people nor the police, government or companies have these possibilities.

1.2 What about intrusion?

1. Break into a system
2. Destroy the log files
3. Install rootkit
4. Find another system and start with 1.

What do you need Tor for then?

Neither normal people nor the police, government or companies have these possibilities.

→ That sucks.

2 The problem

In most public networks, such as the Internet, the connection data reveals who is communicating with whom. This leads to connections being trackable and thus users being identifiable. Encryption doesn't help here, because it does not encrypt this routing data.

Why is this a problem?

- Suppression of sensitive communication
- Censorship
- Infringements on privacy
- Industrial spying
- Changing the outcome of polls due to people being oppressable and corruptable
- Impossibility to do secret elections
- No communication between classified entities is possible

Doesn't *dial on demand* protect the users here?

- You can still tell that a connection was coming from a dial-up line, a certain provider and a certain country.
- Providers are required to keep the data and give it to authorities, while they aren't always well-informed and tend to give it out in cases of presumed copyright infringement too.

3 Requirements to a solution

A solution to this problem must thus meet the following requirements:

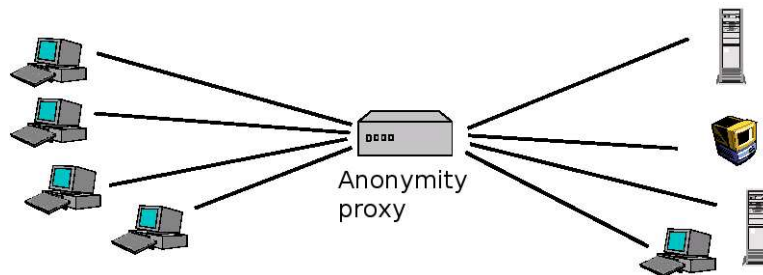
- You must transport traffic for other people to protect yourself, so you can't tell which traffic belongs to which user
- You don't want to trust a single entity (*distributed trust*)
→ The security of the approach is based on the diversity and distributed nature of the network.
- Sender and recipient must be protected against each other (they mustn't know anything about each other)
→ channel and data anonymity
- Someone watching from outside mustn't be able to find out which communication partners belong together
- Someone watching from *inside* mustn't be able to find out which communication partners belong together
→ channel anonymity

It is to be noted here that anonymous authenticated connections can make perfect sense, since for the first time ever they provide real pseudonymity. The authentication must be a property of the data, because the channel doesn't provide any important properties, e.g. the IP may change all the time during a session.

A scenario where anonymous authenticated connections would make sense is when talking to your physician. He will know he's talking to you, and he would be the only one who needs to know that.

4 Prior approaches

4.1 Proxies



In this approach, a provider uses one or more proxies in a row to forward data for the user, who therefore doesn't connect directly to the target host. This works quite well for web connections (also SSL, TLS,

SSH and other low-cost symmetric cryptography systems). Examples would be **The Anonymizer** or **Jap**.

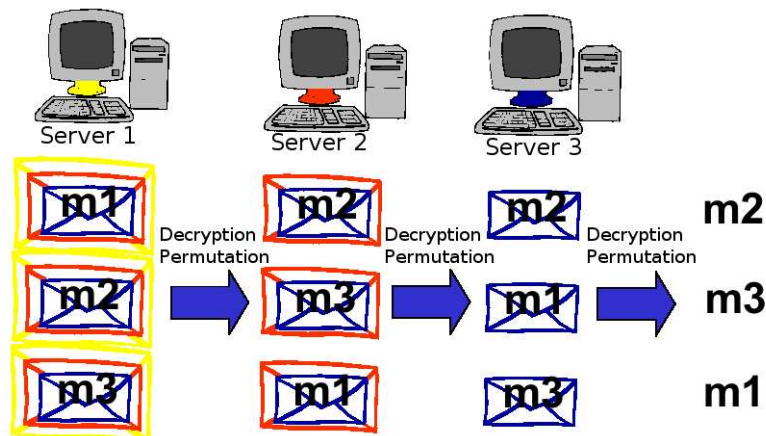
Advantages:

- Easy to configure
- Focusses a lot of traffic to one point and thus makes traffic analysis more difficult

Disadvantages:

- Hardly scalable (You cannot really do load balancing because the traffic load itself is the problem)
- In big companies that use round robin mechanisms to balance the load across multiple proxies, the IPs for the same web connection may vary
- Single point of failure for legal requests, attacks or compromises

4.2 Mixes



A mix decrypts data in a stream hull by hull, whereas he mixes up the order of the messages in the communication channel, so it's impossible to tell in the end which plaintext belongs to which ciphertext.

According to the Chaum model (1981) a client encrypts a message for a mixer chain 3 times with the public keys for 3 different hosts, so the following formula applies:

$$C(p) = E_{PK1}(E_{PK2}(E_{PK3}(p)))$$

where C is the resulting ciphertext, p the plaintext and $PK1$, $PK2$ and $PK3$ the corresponding public keys of host 1, 2 and 3, which were chosen as mixer chain. Every of the 3 servers will then only be able to decrypt the outer hull of the message, so the security is provided as long as at least one of the servers permutes the messages as required.

Advantages:

- Data and channel anonymity are assured
- One honest server preserves the security

Disadvantages:

- Interactivity is rather slow (the connection is laggy)
- Focus of mixer chains lies on high-latency protocols such as email
- Every level of messages requires (expensive) public key cryptography
- No user-friendly implementations are available

4.3 First generation Onion Routing

In this model, a router *throws* an Onion package (see next section) into the network. This so-called *onion* is being handled by a row of routers and finally exits the network again, arriving at the target host as an usual data packet. However, a couple of problems haven't been thought about:

- There was no way yet to classify routers as trustworthy
- Every packet offered a new opportunity to get more information about the sender
- Still uses expensive public key cryptography

5 The principle of Onion Routing

5.1 History

The Onion Router (Tor) implements the *2nd Generation Onion Routing specification*. The main goal was hereby to have the advantages of both mixes and proxies and to reduce expensive public key cryptography to a minimum. The design looks like this:

- Establish circles using (expensive) public key cryptography
- Move data using (cheaper) symmetric cryptography
- Distribute the trust, as done with mixes

5.3 Beware of the DNS!

A severe problem for the anonymity of Tor users is being imposed by "escaping" DNS requests. Only the *Socks4a* and *Socks5* protocol support remote resolution of hostnames, but some clients, such as **Mozilla Firefox**, still resolve the hostname to an IP like for *Socks4* and subsequently connect to that IP.

If some user makes a DNS request for e.g. **www.google.ch**, and then establishes an "anonymous" connection through the Tor network, it's probably pretty obvious who he's connecting to.

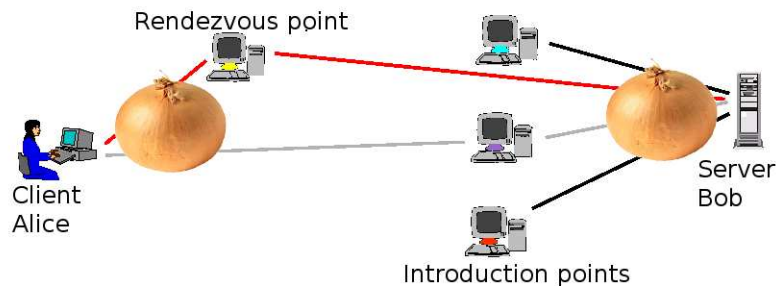
5.4 So who are these nodes?

For the people looking for Tor nodes, Tor has a fairly decentralized directory service. Right now, there are 2 directory servers in the network (**moria1** and **moria2**). The public keys for these servers are part of the Tor source code, so they're preshared and don't have to be exchanged.

The directory itself is decentralized. Every Tor router has a cached copy of the directory which is digitally signed by either of the **morias**, so you can request the directory from any server and still be sure of the validity of the directory.

So it's up to the directory servers to decide whether or not a node may join the network. This prevents the *Sibyl attack* without putting humans into the workflow. At the same time, taking over the network is made very hard because you cannot trivially join an arbitrary number of nodes into the network.

5.5 Hidden services



The idea of hidden services is that **Alice** may connect to **Bobs** service without knowing who Bob is, where his server is located and what his IP address is.

Hidden services are

- Reachable from anywhere where a connection *to* the Internet is possible (even through NATs or HTTP proxies)
- Censorship resistant
- Resistant to distributed Denial of Service attacks
- Resistant to physical attacks (because you don't find them)

If **Bob** decides to offer a hidden service, he just enters the corresponding port mappings into his *torrc*. The Tor client then builds circles to a number of *introduction points* and registers its service, identified by the hash of his public key, along with the introduction points in the directory.

Now when **Alice** decides to connect to **Bob**, she first looks up the hash of his public key in the directory. She chooses one of the introduction points and tells it the nickname of any random Tor router which she chose as *Rendezvous point*. She also sends some kind of authorization along with that. **Bob** then decides whether he wants to talk to **Alice** and builds a circle to the *rendezvous point*. **Alice** builds a circle to it, too, and the *rendezvous point* forwards the data between them.

6 Problems

One of the biggest problems Tor users are facing is acceptance. The english Wikipedia for example refuses to take Tor exit nodes off the IP blocklist, so it's basically impossible to edit articles anonymously. However, the french and german Wikipedia have proven to be very open to Tor, and even asked administrators to have a look into the Tor directory before blocking IPs.

The reasons they brought up were that, apart from the fact that they would keep editors who are concerned about anonymity, that insiders might leave relevant information about interesting topics on Wikipedia, or bring in controverse opinions. It has also been said that a medium which is focused on discussion must be able to deal with the disadvantages of anonymity in order to be able to profit from its advantages.

However, apart from the acceptance aspect, there are a couple of technical problems with Tor:

- An attacker who has control over large parts of the Internet can use timing attacks in order establish correlations between the users sending traffic and the corresponding data that is sent.
- Since Tor only changes its circuits every 10 minutes, it's possible to establish correlations between nonymous and anonymous data being transferred over one exit node.

- An attacker who runs a router inside the Tor network may be able to locate a hidden service running on a non-router by connecting to it until he's chosen as the last node before the service (before a node that doesn't turn up in the directory).

7 Design questions

The following design questions regarding Tor aren't solved yet:

- Padding
- GUI vs. no GUI? (An UI design contest is going on!)
- AS level paths and OSPF like algorithms?
- Random path lengs ($3 + x|x > 0$)
- China?

8 Facts and numbers

- The Tor network is running without interruption since October 2003
- 320 available nodes in 32 countries
- Volunteer-based (other than e.g. the Freedom Network)
- Around 50'000 users (But who can tell? It's an anonymity system!)
- Nodes move between 1 and 100 GB of traffic per day (penrose.thundrix.ch: 16)
- The Tor network has never been down entirely
- Simple client and server in one program of around 40'000 lines of code
- Runs as normal user, doesn't need root access or special permissions and behaves in a BSD jail
- Flexible exit policies enable you to decide what hosts and port a host may connect to if used as an exit node
- There are Tor servers on X-Boxes
- There are Tor servers on Linksys routers

9 Installing Tor

This is a description of the most simple Tor installation on a local computer that cannot be used by other computers in the local network.

For this method you only have to install or compile the corresponding Tor package for your operating system:

Debian Linux, Ubuntu Linux, Debian *BSD, GNU Hurd:

```
% apt-get install tor
```

NetBSD, DragonFlyBSD, Thundrix Linux:

```
% cd /usr/pkgsrc/net/tor && make install
```

or

```
% pkg_add tor
```

FreeBSD, OpenBSD, MirBSD:

```
% cd /usr/ports/net/tor && make install
```

If your system doesn't have any adequate package management software, you can also install pkgsrc and follow the instructions for NetBSD.

After that you only need to start Tor (example: NetBSD):

```
% /etc/rc.d/tor start
```

Now you can tell your applications to use **localhost:9050** as a Socks4a or Socks5 proxy, and it should use Tor.

9.1 Privoxy for web browsing

Since **Mozilla Firefox** appears to be vulnerable to the DNS attack, it is recommended to install Privoxy if you want to browse the web with Tor. Also, Privoxy filters data from the headers the browsers sends to the website, so you're leaving less hints about your identity.

The installation of Privoxy works not quite unlike that of Tor:

Debian Linux, Ubuntu Linux, Debian *BSD, GNU Hurd:

```
% apt-get install privoxy
```

NetBSD, DragonFlyBSD, Thundrix Linux:

```
% cd /usr/pkgsrc/www/privoxy && make install
```

or

```
% pkg_add privoxy
```

FreeBSD, OpenBSD, MirBSD:

```
% cd /usr/ports/www/privoxy && make install
```

Once the installation has finished, you should edit the `/etc/privoxy/config` in the following way:

- Add a line reading: `forward-socks4a / localhost:9050 .` (Note: up to and including the dot)
- Comment out the line reading `logfile logfile`
- Comment out the line reading `jarfile jarfile`

Since we're trying to protect our privacy, it's probably safe to assume that we don't want to keep log files.

Then you will need to restart privoxy:

```
% /etc/rc.d/privoxy restart
```

Now you just have to tell your browser to use the HTTP proxy `127.0.0.1` on port `8118`.

As a first test you should try to access the Hidden Wiki. If it works, everything is set up just fine. If it doesn't, you probably have the DNS problem, or the connection doesn't run over Tor at all.

9.2 Socat for SSH

In order to have SSH run over Tor, you can use the program `socat`, for example. To do so, you need to install it first, of course:

Debian Linux, Ubuntu Linux, Debian *BSD, GNU Hurd:

```
% apt-get install socat
```

NetBSD, DragonFlyBSD, Thundrix Linux:

```
% cd /usr/pkgsrc/net/socat && make install
```

or

```
% pkg_add socat
```

FreeBSD, OpenBSD, MirBSD:

```
% cd /usr/ports/net/socat && make install
```

Finally you need to edit either the `/.ssh/config` if you want a user specific setting, or the `/etc/ssh/ssh_config` if you want a general setting. Whichever file you choose, you have to append the following formulation:

```
Host *
```

```
ProxyCommand socat STDIN SOCKS4A:127.0.0.1:%h:%p,socksport=9050
```

Of course, you can also limit that to the .onion nodes so you can still reach the rest of the net without Tor:

```
Host *.onion
ProxyCommand socat STDIN SOCKS4A:127.0.0.1:%h:%p,socksport=9050
```

Now you can try whether the SSH actually does what it ought to do:

```
% ssh -l anonymous oxdjh4bmutfxv2ba.onion
```

The password is **sarahiscute**, and **Cone** begs the users to keep the vandalism low.

10 Web links

- The official Tor website
- The Hidden Wiki
- Tor FAQ on noreply.org
- Torifying HOWTO - how to force several applications to cooperate
- Privoxy
- Readable Tor directory