# CS294-28 - Network Security - Spring 2008
# Anonymity

Prof. Vern Paxson
Scribed by Bonnie Zhu[*]

**Abstract**

In today's class, we cover the topic of anonymity mainly through the discussion of *Tor*. The focus of this lecture leans towards the design and usage of such systems, instead of conceptual or theoretical notions associated with anonymity. For more details please refer to Tor's website [1].

## 1 Why Anonymity?

There are many occasions at which we'd like to have anonymity. To name a few,

- protection of privacy, such as
  - freedom of speech: journalists, dissidents, whistleblowers;
  - law enforcement (In-q-tel, Nye Kripos);
    - Anonymous tips or crime reporting
    - Surveillance and honeypots (sting operations)
  - (**Not mentioned in lecture**) Socially sensitive communicants: chat rooms and web forums for abuse survivors, people with illnesses;
  - censorship-resistant systems;
- protocol design where malicious router can't be detected;

**Note**: Side info in reality changes the theoretical definition of *anonymity*. You can't be anonymous by yourself but can have confidentiality by yourself, i.e., anonymity loves company.

## 2 How to get communication anonymity?

There are many technical approaches, among which two are extensively used: *Mixes* and *Proxies*.

## 3 The paper – Tor

Dingledine, Mathewson and Syverson [5] present *Tor*, a circuit-based low-latency anonymous communication service. It uses asynchronous and improved *onion routers*, a distributed overlay network design.

Tor combines advantages of mixes and proxies

- Use (expensive) public-key crypto to establish circuits
- Use (cheaper) symmetric-key crypto to move data stream

[*]Disclaimer: The notes is based on lecture minutes and incorporated much content from Tor's website [1]. All mistakes are the scriber's. Please send feedback to bonniez@berkeley.edu.

  - Like SSL/TLS based proxies

- Distributed trust like mixes

The focus of Tor is anonymity of the communication pipe, not what goes through it.

## 3.1 Contribution

- usability: quirky yet doable;
- improvement over onion router: run on edge, low-latency, *telescoping* path-building design;
- scalability: started with 32 nodes and now has 2300 nodes[1].

**Comment**: The paper's thoroughness is as that of a system paper with several pieces subtle.

## 3.2 Core tensions

The major tradeoffs facing Tor are the ease of deployment, acceptably low latency and liability burden.

## 3.3 Model

Tor downloads topology from $K$ servers and builds circuits plus provides hidden service.

It offers service for interactive traffic which requires low-latency[2].

- All service is done on network layer, niche network not application-layer. If one wants application layer instead, then the technique used is application protocol normalizer.
- It's deployable. However, there's a lability issue: traffic coming from you ought to look like belong to you.
- The Tor system, provides a mode that tunnels everything over TCP Port 80 which is often not filtered, given it is usually reserved for HTTP (Web) traffic.
- Exit position can be traced through its associated port number – http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers gives a table of TCP/UDP port number, e.g. Port 465 may be used for Cisco protocol thus for association with attacks targeting routers.
- Exit node activity scales with exit node.
- Perfect forward sequence if its public key is compromised; session keys are used on how to construct them on top of integrity checking.
- Model is stream based instead of data-gram based.

# 4 Another Approach

Ari explained another approach to do anonymization through broadcasting.

Given a group of nodes, everyone has public key. Since everyone is shouting then you won't be able to tell who is really sending the message. It's provably secure.

Over such shared broadcast bus, bandwidth is linear to the number of nodes.

[1]In the paper, the authors suspect *Tor* doesn't scale well.

[2]In contrast, Mixnet such as *Remailer* can tolerant mid to high latency.

## 5 Threat Model

Tor does **not** protect against

- a global passive adversary;
- *traffic confirmation* attacks;
- revelation of who is connected to the network;

*but*

- *timing analysis* attacks;
- attacks capable of generating, modifying, delaying and deleting traffic.

Tor opts for getting security though being highly usable and cheap to operate.

As a result, an adversary who can observe a stream at two different points, can trivially realize it is the same traffic.

## 6 The Tor Design

The main idea in design is the following,

- Traffic is forwarded through several routers and multiply encrypted, with each *Onion Router* OR removing one layer of the encryption.
- In particular, the first (entry) router knows the source of the tunnel, but not its destination, and the last (exit) router knows the destination but not the source.
- Each router reports its IP address, public key, policies about what traffic it will accept,and a bandwidth value that is determined by monitoring the peak bandwidth achieved by the router over a period of time.

The Tor onion routing meshes TCP with TLS, a datagram-based protocol [3]. Each OR maintains a TLS connection to every other OR.

### 6.1 Tor Circuit Setup

When a client's application wants a TCP connection to a given address and port, it asks the *Onion Proxy* (OP), user's local software, to fetch directories, establish circuits and through SOCKS to make the connection.

An OP establishes session key and circuit with *Onion Router 1*. Then the *Proxy* tunnels through that circuit to extend to *Onion Router 2*. Analogously, to *Onion Router 3*. The process is pictorially shown in Fig 1.

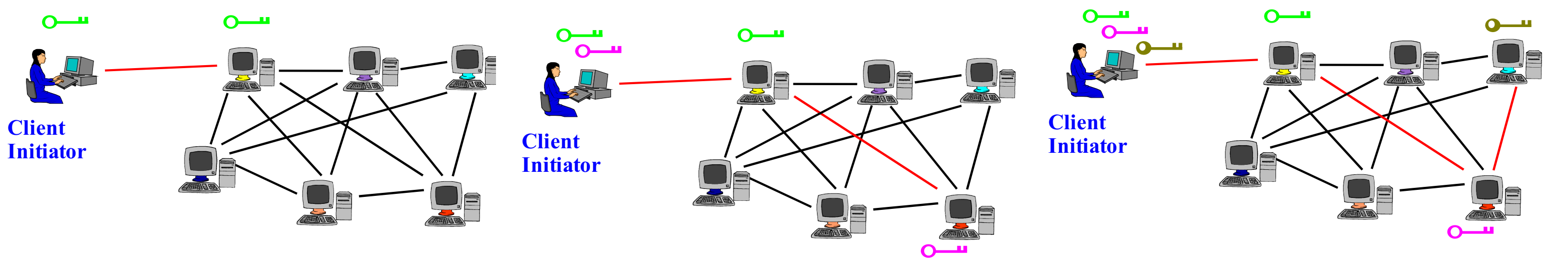

Figure 1: The process of building a circuit Source: [1]

Fig 2 illustrates the overall pictorial view on the process with data flow incorporated .

Now, client applications connect and communicate over Tor circuit, shown in Fig 3.

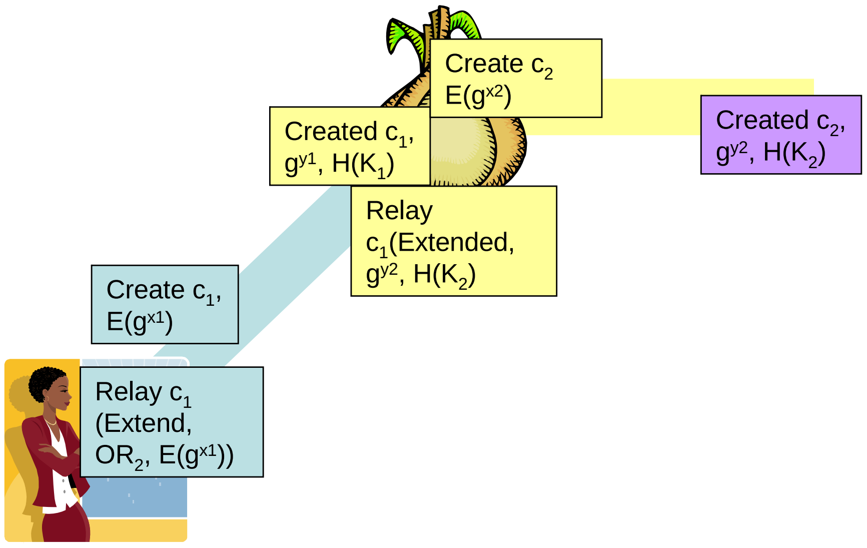
Create $c_2$, $E(g^{x2})$
Created $c_1$, $g^{y1}$, $H(K_1)$
Created $c_2$, $g^{y2}$, $H(K_2)$
Relay $c_1$(Extended, $g^{y2}$, $H(K_2)$)
Create $c_1$, $E(g^{x1})$
Relay $c_1$ (Extend, $OR_2$, $E(g^{x1})$)

Figure 2: Building a circuit Source: [1]

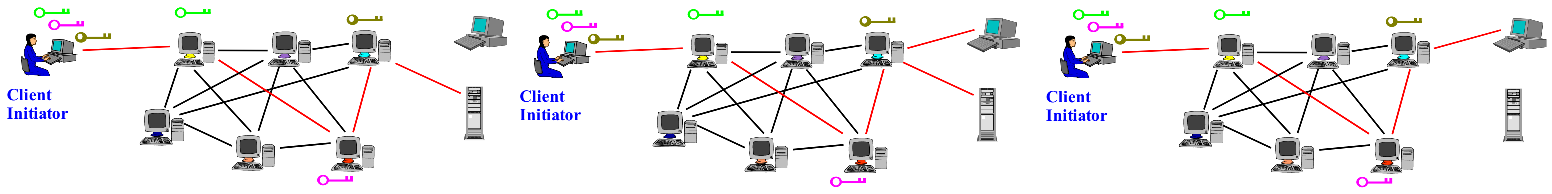

Figure 3: How to use a circuit Source: [1]

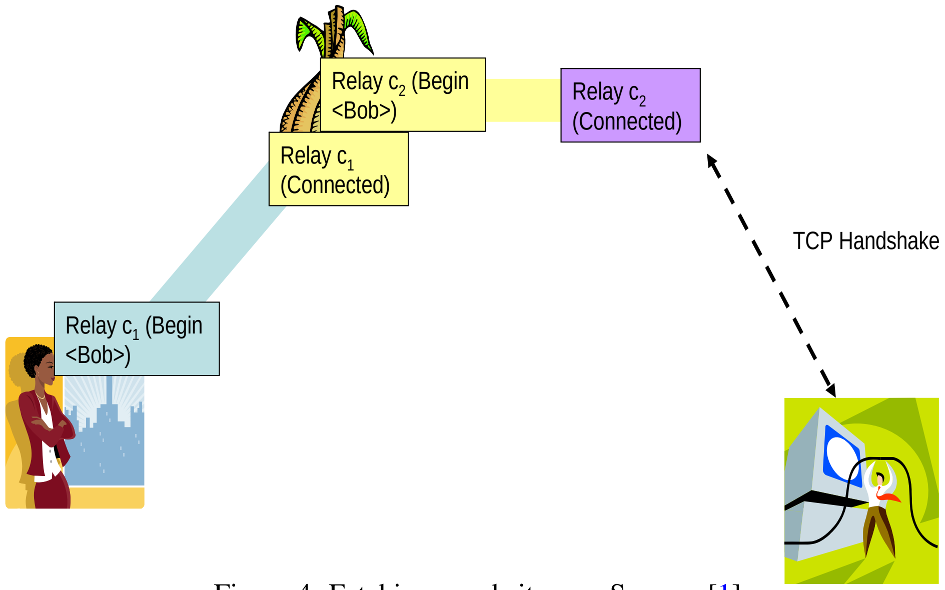

Figure 4: Fetching a website Source: [1]

So the client is ready to fetch a website, as shown in Fig 4. Note the last onion router should get the website's IP address that the client wants to access in order to protect the client's anonymity.

**Note**

- Directory service doesn't scale.
- Anonymity is a protocol building block.

## 6.2 Cells

*Onion Proxy* (OP) uses TCP, fetches directories and creates circuits on the network layer. All data is sent in fixed size cells, see Fig 5.

| 2 | 1 | 509 bytes |
|---|---|---|
| CircID | CMD | DATA |

| 2 | 1 | 2 | 6 | 2 | 1 | 498 |
|---|---|---|---|---|---|---|
| CircID | Relay | StreamID | Digest | Len | CMD | DATA |

Figure 5: Overview of cell structure and the details of relay cell structure Source: [5]

Fig 5 also shows an interesting command: *relay*. The entire *relay* command cell is encrypted or decrypted as the relay cell moves along the circuit, using the 128-bit AES cipher stream. It looks *digest* after decryption.

Fig 6 illustrates the overall data stream diagram on how to build a two-hop circuit and to fetch a web page afterwards.

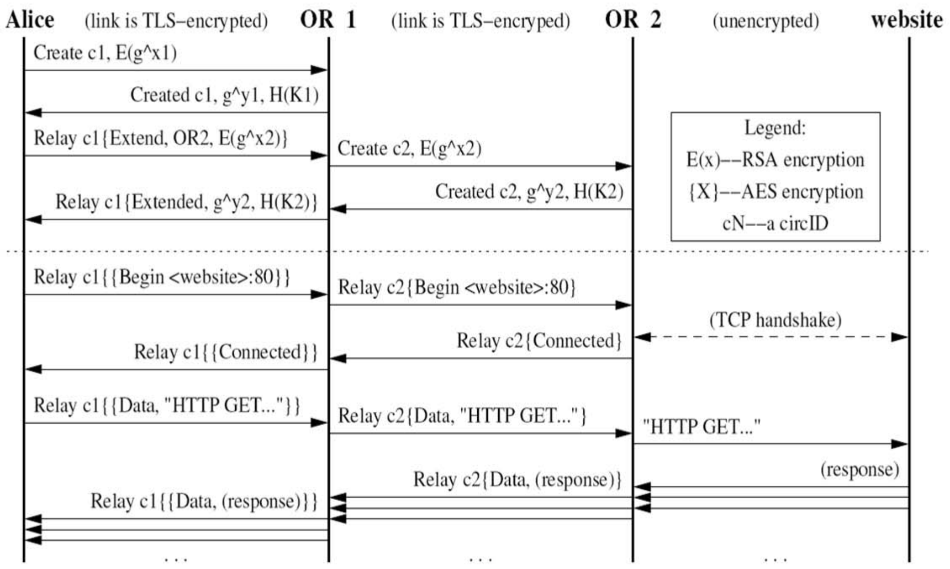

Figure 6: Alice builds a two-hop circuit and begins fetching a web page Source: [5]

Over the Tor network, traffic is forwarded through several routers and multiply encrypted, with each OR removing one layer of the encryption. In particular, the first (entry) router knows the source of the tunnel, but not its destination, and the last (exit) router knows the destination but not the source. Each hop knows only adjacent routers and their associated information but not any of non-adjacent hops. By using such *leaking pipe* mechanism, Tor is able to fend off *traffic analysis*. We don't make any assumption on the change of client.

## 6.3 Crypto

During its initial connection to SOCKS, a client asks to connect to an IP address. The server who has access to DNS sees who just looked him up. Subsequently, DNS lookup would pop up within TOR.

There is a subtlety on crypto in the paper to illustrate a standard pitfall for non-crypto people applying crypto. The old Onion Routing design uses AES cipher stream without integrity checking, thus subject to *malleability attack*. Suppose one can guess the encrypted content, the s/he might also be able to change a padding cell to a destroy cell; change the destination address in a *relay begin* cell to its own webserver and so on[3].

## 6.4 Attacks

All the attacks boil down to the issue of action distinguishability, e.g. which machine corresponds to which IP and which MAC address matches that IP. MAC address includes vendor code, which may be relatively easier to trace.

A related research topic is *Trace Anonymization*.

## 6.5 Hidden Service and Rendezvous Points

Tor accommodates receiver anonymity by allowing *Location Hidden Services*. Location hidden service leverage *Rendezvous Points*(RP).

The design goals for location hidden services are

- Access Control
- Robustness
- Smear-resistance
- Application transparency

If Alice wants to access Bob's service anonymously, she must first connect to the lookup service via Tor.

As illustrated in Fig 7, she first Alice chooses an OR as the RP for her connection to Bob's service. She builds a circuit to the RP and gives it a randomly chosen "rendezvous cookie" to recognize Bob.

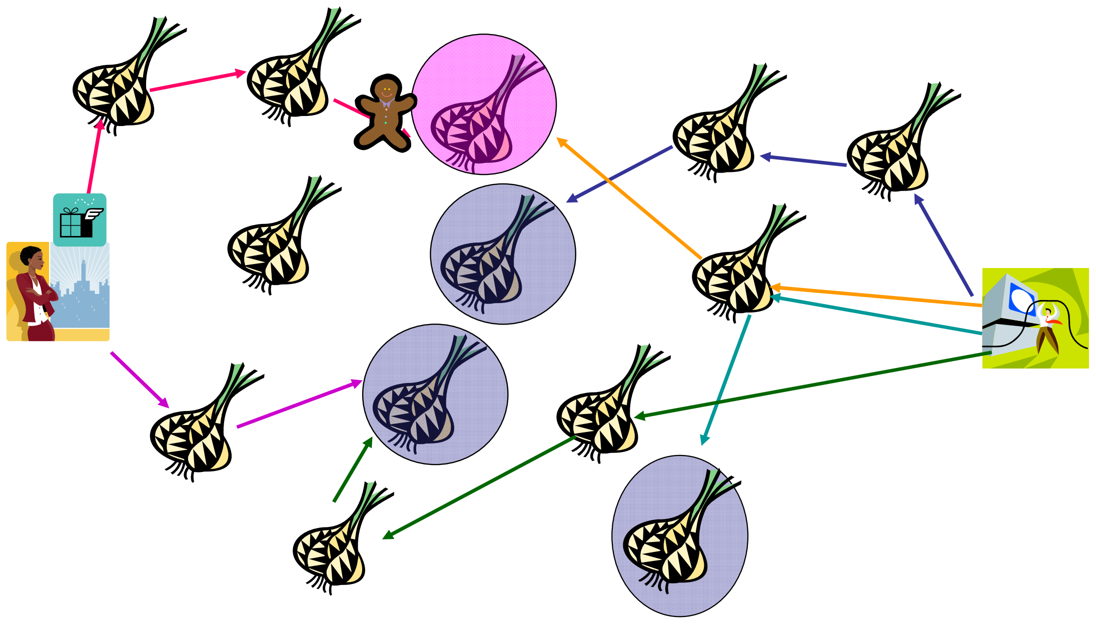

Figure 7: Creating and connecting to a Location hidden service Source: [1]

[3]Keep in mind, TCP checksum alone is too lousy to be effective against such malicious activities.

**Note**

- Although RP connects Alice's circuit to Bob's, it can't recognize Alice, Bob, or the data they transmit.
- Alice can connect to Bob's server without knowing where it is or possibly who he is.
- Alice's access to Bob's service is anonymous too.

**Not fully covered in the lecture** Location hidden service can provide servers that

- Are accessible from anywhere
- Resist censorship
- Require minimal redundancy for resilience in denial of service (DoS) attack
- Can survive to provide selected service even during full blown distributed DoS attack
- Resistant to physical attack (you can't find them)

## 7 Concluding Remarks

Anonymity *does* matter. Anonymous communication research and provision are increasingly relevant. Systems that are the direct result of this research, like Tor, JAP[4] and Mixminion [2], are deployed and used to protect the privacy of thousands of people.

## References

[1] Tor Project: http://www.torproject.org/index.html.en

[2] G. Danezis, R. Dingledine and N. Mathewson *Mixminion: Design of a Type III Anonymous Remailer Protocol*, in Proceedings of IEEE Symposium on Security and Privacy, Berkeley, CA, (2003)

[3] T. Dierks and C. Allen. *The TLS Protocol Version 1.0.* IETF RFC 2246, January 1999. http://www.ietf.org/rfc/rfc2246.txt

[4] R. Dingledine *Tor: Anonymous Communications for the United States Department of Defense...and you*, http://freehaven.net/ arma/wth1.pdf

[5] R. Dingledine, N. Mathewson, P. Syverson, *Tor: The Second-Generation Onion Router*, USENIX Security 2004

[4]http://anon.inf.tu-dresden.de/