Dingledine and buddies 2004; how to read computer science papers and stop worrying about the future.

Camille Akmut

Abstract

One of many more ways in which computer science can be conducted. We guide the reader through Dingledine et al.'s 2004 paper on Tor.

# Make use of all resources

—–

Is there a computer scientist that you like, or think is cool? They explain things well, or in interesting ways? Have they written something about the topic of the paper you are trying to read? What of textbooks?

Gather those before starting. We used :
- the accessible *Understanding Cryptography*, *Serious Cryptography*,
- one paper co-written by Ian Goldberg,
- a Computerphile / Nottingham University video,
- two average Networking textbooks (the well-known ones).

—–

We begin with the title :

> *Tor: The Second-Generation Onion Router*

So, there were other generations of this before Tor? Correct. The perhaps best introduction to the topic has been given in Kate, Zaverucha and Goldberg 2010, where they write :

> Over the years, a large number of anonymity networks have been proposed and some have been implemented. Common to many of them is *onion routing* [Reed et al. 1998], a technique whereby a message is wrapped in multiple layers of encryption, forming an *onion.* As the message is delivered via a number of intermediate *onion routers* (abbreviated ORs, also called *hops* or *nodes*), each node decrypts one of the layers, and forwards the message to the next node. This idea goes back to Chaum [1981] and has been used to build both low- and high-latency communication networks.
>
> (...) the original Onion Routing project [Goldschlag et al. 1996; Reed et al. 1998; Syverson et al. 2000] (...) was superseded by Tor (...)

We can then proceed to the abstract.

> *We present Tor, a circuit-based low-latency anonymous communication service.*

"*Circuit-based*" sounds perhaps like it might mean more than it does at first, but should not be attributed or given too much meaning only because it is found here, in this context. A circuit, according to any common dictionary, is a "path" or a "route".

A topic-specific definition, description for this is found just later in the text :

> Clients choose a path through the network and build a *circuit*, in which each node (or "onion router" or "OR") in the path knows its predecessor and successor, but no other nodes in the circuit.

Pictures are much better than words at this; and here the best illustration so far has been given by Computerphile :

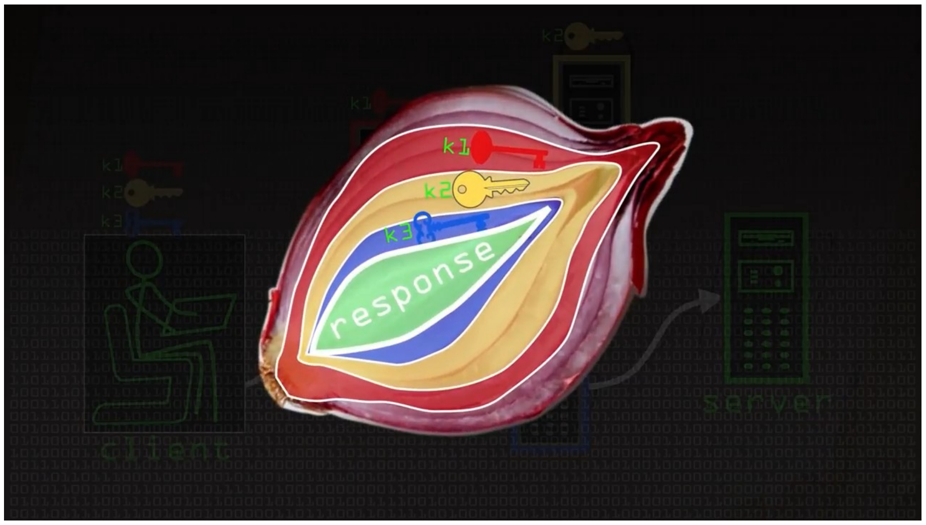

Figure 1: A message contained in 3 layers of encryption. (Computerphile)

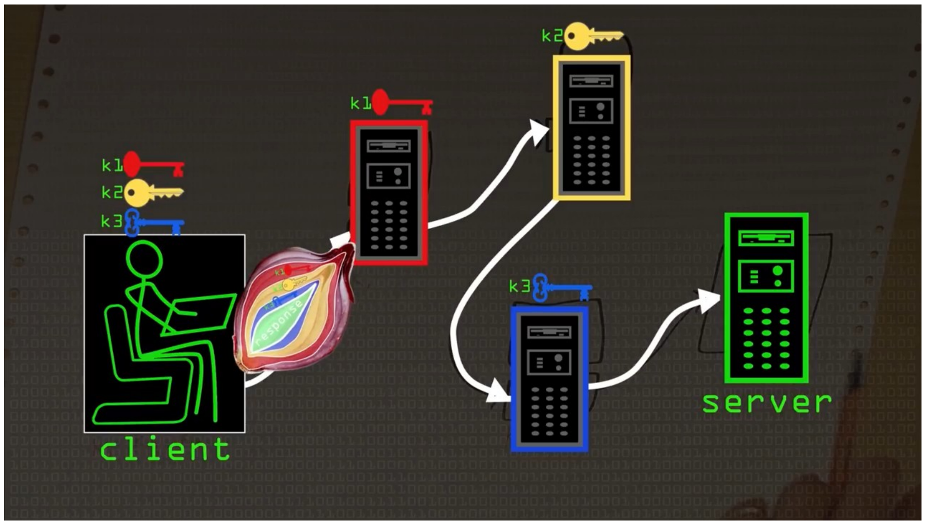

Figure 2: The message (i.e. response) travels back from the server to the user.

Hence the whole onion terminology : onion routing, onion router, etc.

# “Neither a beginning nor an end”

To find more clues about “*low-latency*” we need to skip forward a bit. This might seem counter-intuitive, but computer science literature isn’t really a novel : perhaps better described, used and enjoyed as a ‘Choose Your Own Adventure’-type of deal.

On the second page of this paper, Dingledine et al. explain :

> Modern anonymity systems date to Chaum’s **Mix-Net** design [10]. Chaum proposed hiding the correspondence between sender and recipient by wrapping messages in layers of public-key cryptography, and relaying them through a path composed of “mixes.” Each mix in turn decrypts, delays, and re-orders messages before relaying them onward.
>
> Subsequent relay-based anonymity designs have diverged in two main directions. Systems like **Babel** [28], **Mixmaster** [36], and **Mixminion** [15] have tried to maximize anonymity at the cost of introducing comparatively large and variable latencies. Because of this decision, these *high-latency* networks resist strong global adversaries, but introduce too much lag for interactive tasks like web browsing, Internet chat, or SSH connections.
>
> Tor belongs to the second category: *low-latency* designs (...)

High-latency : slow. Low-latency : fast.
Mix : (precursor terminology for) node.
Let’s move forward.

> This second-generation Onion Routing system addresses limitations in the original design by adding perfect forward secrecy, congestion control, [long list] (...)

In this long list of advantages, (perfect) “*forward secrecy*” comes first and it is indeed of importance.

But, one is hard pressed to find much help in many textbooks of this literature : these gentlemen and gentlewomen prefer concentrating on the maths.

Many textbooks, of which we will spare the authors of publicity, disregard this notion even under its jargon synonym “backtracking resistance”.

> Backtracking resistance (also called *forward secrecy*) means that previously generated bits are impossible to recover, whereas prediction resistance (*backward secrecy*) means that future bits should be impossible to predict.[1]

In *Understanding Cryptography* a definition can be gotten too[2], but much better are those found on the ’net. We are not sure what this says about the quality of computer science textbooks in general.

[1] *Serious Cryptography*, chapter 2.
[2] 13.2.3 Remaining Problems with Symmetric-Key Distribution.

> On top of the usual confidentiality and integrity properties of HTTPS, forward secrecy adds a new property. If an adversary is currently recording all (...) users' encrypted traffic, and they later crack or steal [the server]'s private keys, they should not be able to use those keys to decrypt the recorded traffic.[3]

# Understanding the basics is more than 99%

> *Onion Routing is a distributed overlay network designed to anonymize TCP-based applications like web browsing, secure shell, and instant messaging.*

The details of which, both protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), of the transport layer[4], can be found in any textbook containing "Networks" or "Networking" as part of its title :

> The UDP protocol provides a connectionless service to its applications. This is a no-frills service that provides no reliability, no flow control, and no congestion control.[5]

It should hopefully make sense now why it is not employed by Tor.

If not, while continuing reading one will stumble on passages such as this one, which reiterate previously encountered concepts :

> **6 Other design decisions**
>
> **6.1 Denial of service**
>
> *Providing Tor as a public service creates many opportunities for denial-of-service attacks against the network. (...) flow control and rate limiting (discussed in Section 4.6) prevent users from consuming more bandwidth than routers are willing to provide*

# Actionable knowledge

We do not read computer science papers, and later write them ourselves, as the past time or respite of the cultivated and gentle mind.

To change our minds, to change our lives.

—

BitTorrent also works over TCP (some things like trackers can use UDP however[6]).

It is out of respect for other users that the Project asks to not pair Tor and BitTorrent, as Roger Dingledine's answer made clear in 2013 :

[3]https://blog.twitter.com/engineering/en_us/a/2013/forward-secrecy-at-twitter.html
[4]Tanenbaum and Wetherall, *Computer Networks*, "Introduction".
[5]Kurose and Ross, *Computer Networking.* 1.5 "Protocol layers and their service models"
[6]https://blog.torproject.org/bittorrent-over-tor-isnt-good-idea

> The blog post is still accurate. Please don't do it.
>
> Sending your BitTorrent traffic through the Tor network would overload it even more. It isn't designed to handle such things – the Tor network has much less capacity than it has users wanting to use it. And since it's zero-sum, every person trying to BitTorrent over Tor means many more people in Syria who can't get to their Facebook pages.[7]

This might change in the future, who knows.

# Skip what you don't understand

Computer science should be fun. If it isn't you're doing something wrong. Skip what you don't understand, and come back to it later.

Bibliography

Kate, Aniket, Zaverucha, Greg and Goldberg, Ian. 2010. "Pairing-Based Onion Routing with Improved Forward Secrecy"

Computerphile. 2017. "Onion Routing". QRYzre4bf7I

Cohen, Bram. 2017 [2008]. "The BitTorrent Protocol Specification". http://www.bittorrent.org/beps/bep_0003.html"

[7] https://tor.stackexchange.com/questions/64/how-can-bittorrent-traffic-be-anonymized-with-tor/68#68