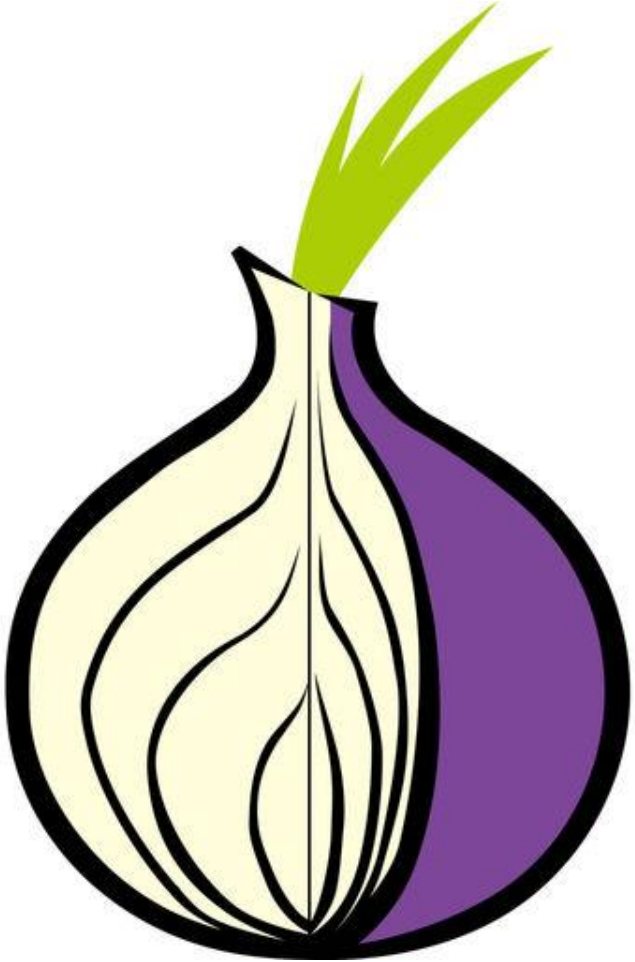
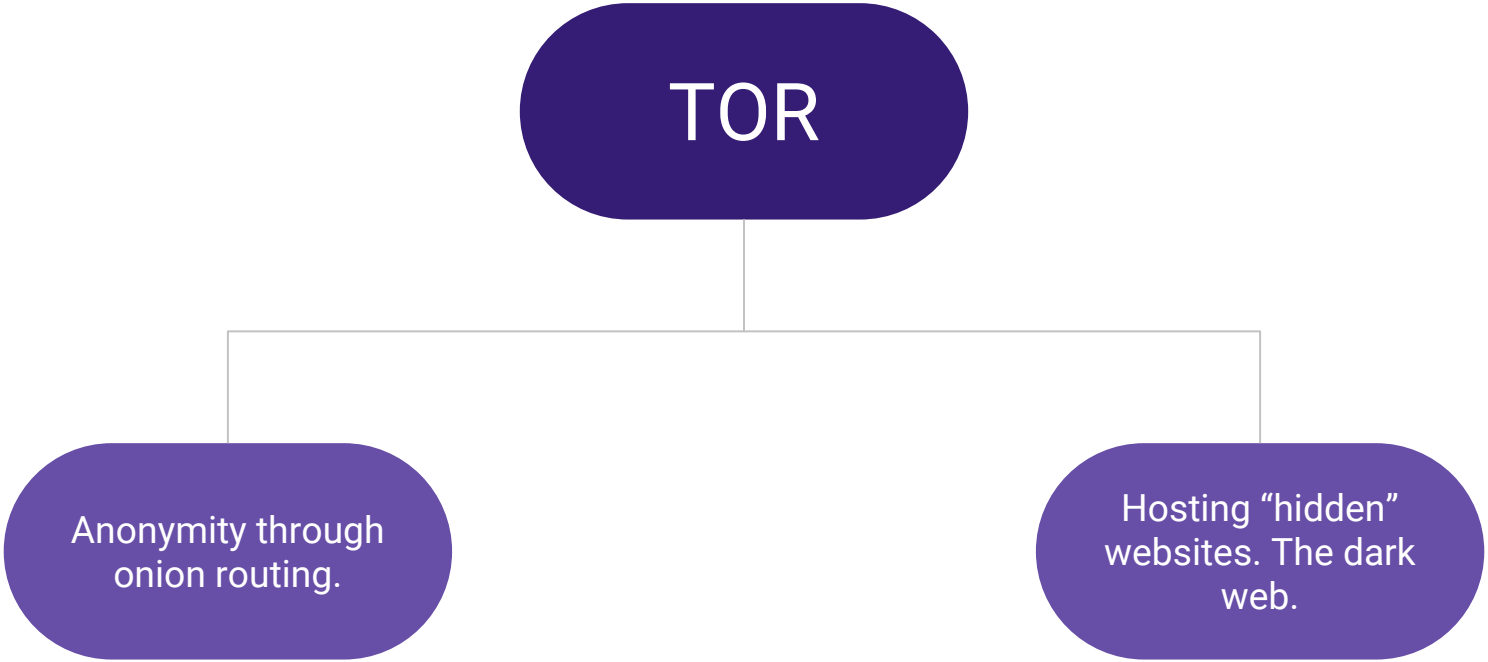
# TOR

Onion routing and hidden services




TOR
Anonymity through onion routing.
Hosting “hidden” websites. The dark web.

# Why do people use TOR?

--Doxing Prevention

--Whistleblowers

--Political Views

--Sensitive Topics

--Censorship

--Bypass Surveillance

--Law Enforcement

--Cheese Pizza

"https://facebookcorewwwi.onion/".
http://3g2upl4pq6kufc4m.onion/

# Why is encryption not enough?

- Encryption hides the payload. Headers are left exposed. Both to the receiver and Eve.

- Tor protects against traffic analysis, which reveals a lot of information (e-commerce price discrimination based on country)

# Alternatives. VPN?

- VPNs reveal the exact amount and timing of communication.
- ISPs can keep logs of all of your network traffic.
- Censorship not escaped
- ProxyChaining
  - Entry Node still knows who the exit node is

# What is Onion Routing?

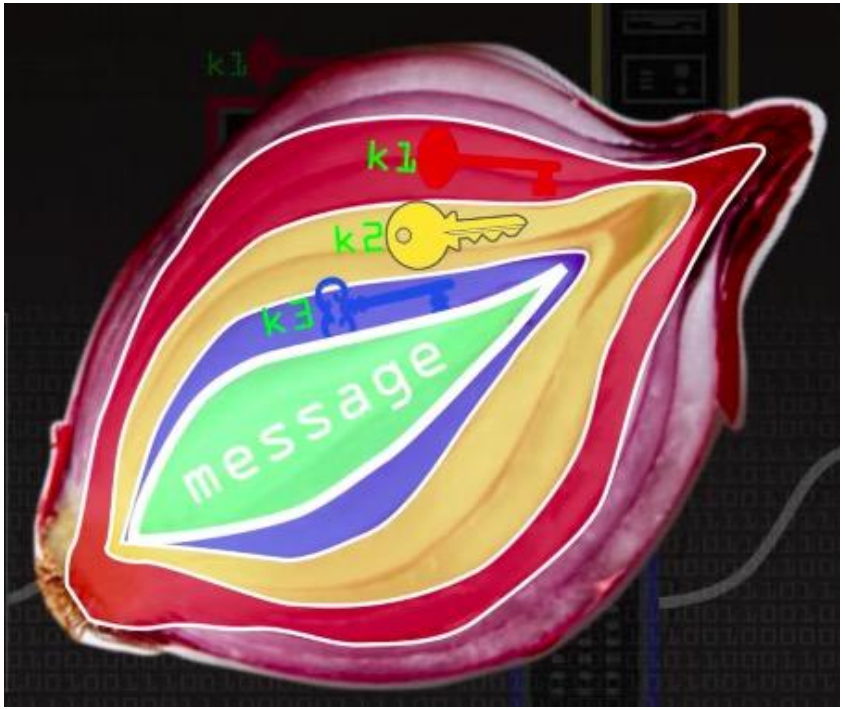
- Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays.

- Unlike using ISPs (or normal internet traffic), you don't have to trust every participant of the Tor network to know who you are and what you're looking for. WHY? Since everyone in the network only knows what's behind them and what's after them.

- Traffic is routed through multiple nodes. This makes it a little bit more difficult to trace the route but it is still very traceable. Why?

# Onion Routing continued

- The smart bit is using layered encryption so every node can only decrypt part of the message.



# Main Players:*

1. **A CLIENT's local software aka onion proxy:**
   a. fetches directories
   b. establishes circuits across the network
   c. handles connections from user applications

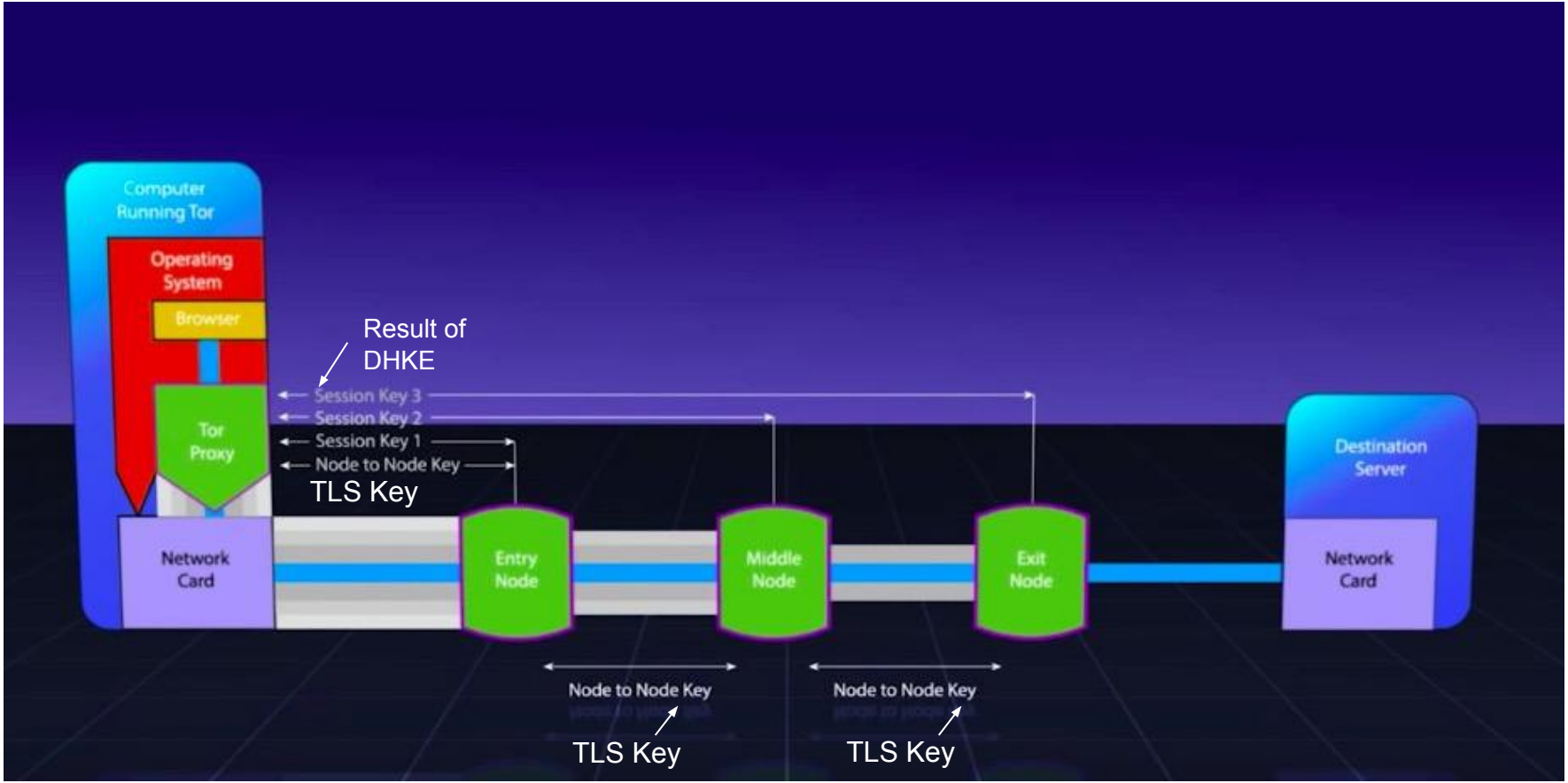
2. **A NODE (ONION ROUTER) maintains a long-term identity key and a short-term onion key:**
   a. Identity key:
      i. sign TLS certificates
      ii. sign the node's *router descriptor* (a summary of its keys, address, bandwidth, exit policy, etc)
      iii. sign directories (if node is a directory server)
   b. Onion key: (***RSA Private key!)***
      i. decrypt requests from users to set up a circuit and negotiate ephemeral (short-term) session keys

3. **Packets are called CELLS. Each cell is 512 bytes.**

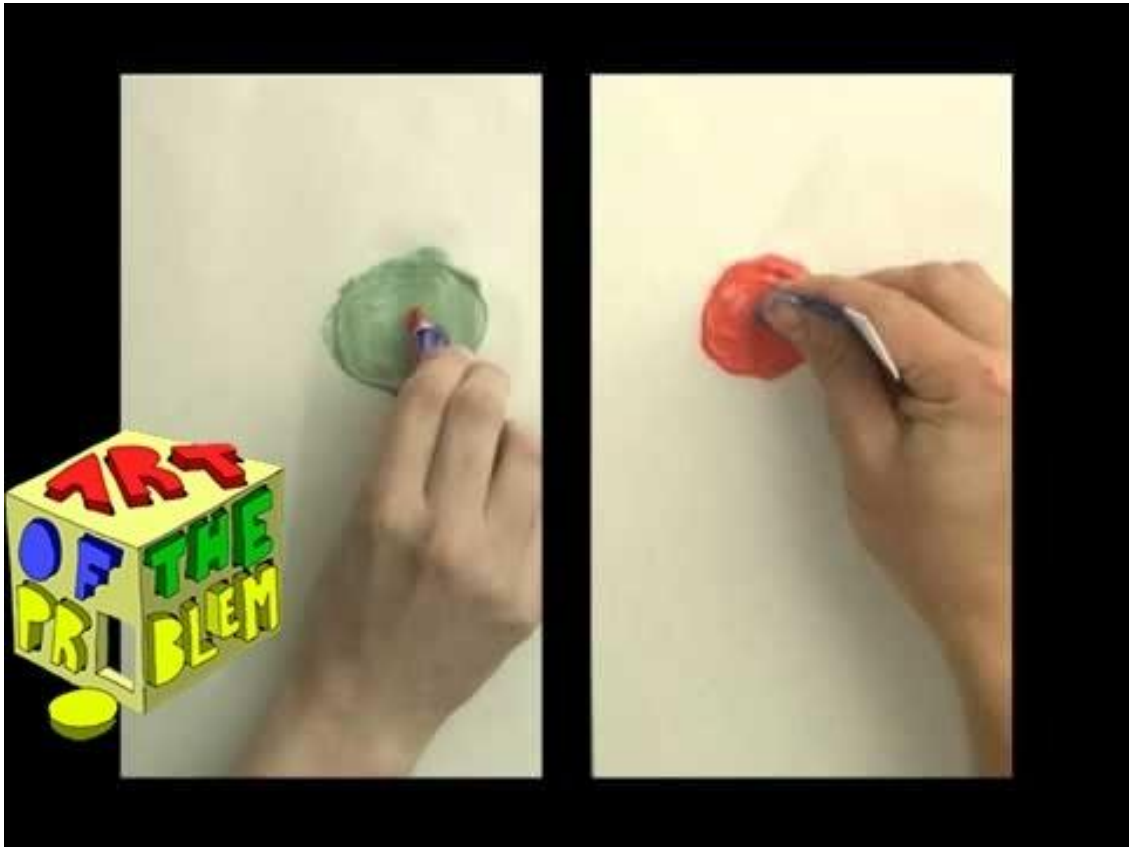
# Main Protocols:*

1. **TLS:**
    a. Layer on top of all communications.
    b. Why? So Eve does not know what is happening (Is a key exchange happening? Is a normal message being sent? Eve doesn't know).
2. **RSA:**
    a. Only when setting up session keys to create a circuit.
    b. Why? So that we can talk to nodeX (through other nodes) and have ONLY nodeX understand what we're saying.
3. **Ephemeral DHKE (ephemeral = short-term keys, change often):**
    a. Only when setting up session keys.
    b. Why make session keys, why not just use RSA? Can't send large messages using RSA!
    c. Why Ephemeral? So we have "Perfect Forward Secrecy" i.e. if RSA private key is ever compromised in the future, the attacker can't decrypt previous messages
4. **AES**
    a. To actually encrypt the data!
    b. We give AES the session key that DHKE set up for us

*

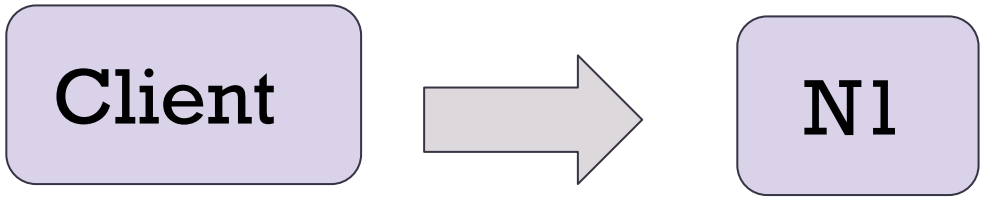

Computer Running Tor
Operating System
Browser
Tor Proxy
Network Card
Result of DHKE
Session Key 3
Session Key 2
Session Key 1
Node to Node Key
TLS Key
Entry Node
Middle Node
Exit Node
Destination Server
Network Card
Node to Node Key
Node to Node Key
TLS Key
TLS Key

# Diffie-Hellman Key Exchange



# Circuit Construction Steps*

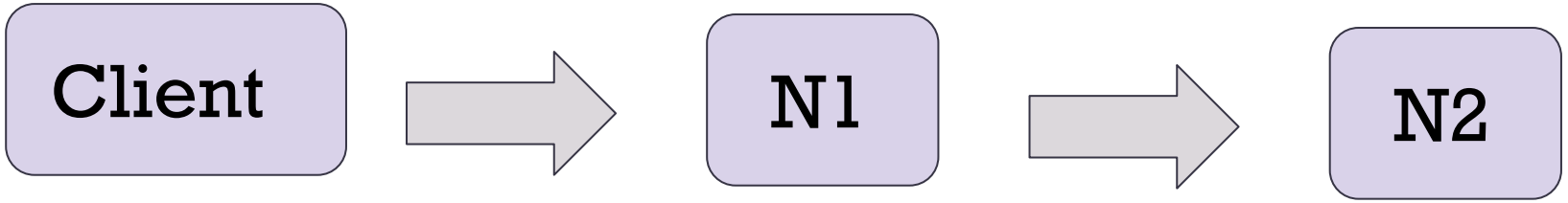
Client, C, wants to connect to a server, S through the TOR network. Remember packets in TOR are "cells". All of this happens within a TLS layer.




1. C sends a "create" cell to the first node N1 (of C's choice). This contains the first half of the DHKE ($g^x$) encrypted using N1's onion key (RSA private key).
2. N1 sends a "created" cell to C. This contains the second half of DHKE ($g^y$) along with a hash of the negotiated key K1=$g^{xy}$

# Why bother have N1 send a hash of the key back to C? *

To make sure that Mallory can not claim to be N1 and spoof this connection




1. C sends a "relay extend" cell to the first node N1. This contains the address of the next node chosen by C, N2, and the first half of the DHKE ($g^{x2}$) encrypted using N2's onion key. Each node keeps track of its next hop, so the client does not share this info again.

2. N1 copies this encrypted half-handshake ($g^{x2}$) into a "create" cell and passes it on to N2.

3. N2 replies with a "created" cell to N1. This contains the second half of DHKE ($g^{y2}$) along with a hash of the negotiated key $K2=g^{x2y2}$

4. N1 now wraps the "created" message into a "relay extend" cell and sends that to C. This is repeated for a selected N3.

# Does N2 know who C is?

No, it just knows that it's creating a symmetric key with someone but does not know who that is.

# Idea: can we use a public key to encrypt traffic returning to C? *

No, that makes C identifiable to all the nodes.

# Using the Circuit

*

1. C sends a "relay" cell to N1 after encrypting the cell payload (that is, the relay header and payload) with K3, K2, then K1.
2. N1 decrypts the relay header and payload with the session key, looks up the node for the next step in the circuit (N2) and sends the decrypted relay cell to N2.

- When N3 later replies to C with a relay cell, it encrypts the cell's relay header and payload with the single key it shares with C, and sends the cell back toward C along the circuit. Subsequent nodes add further layers of encryption as they relay the cell back to C.

Note: Circuits are built preemptively in the background to avoid delays

# Summary:

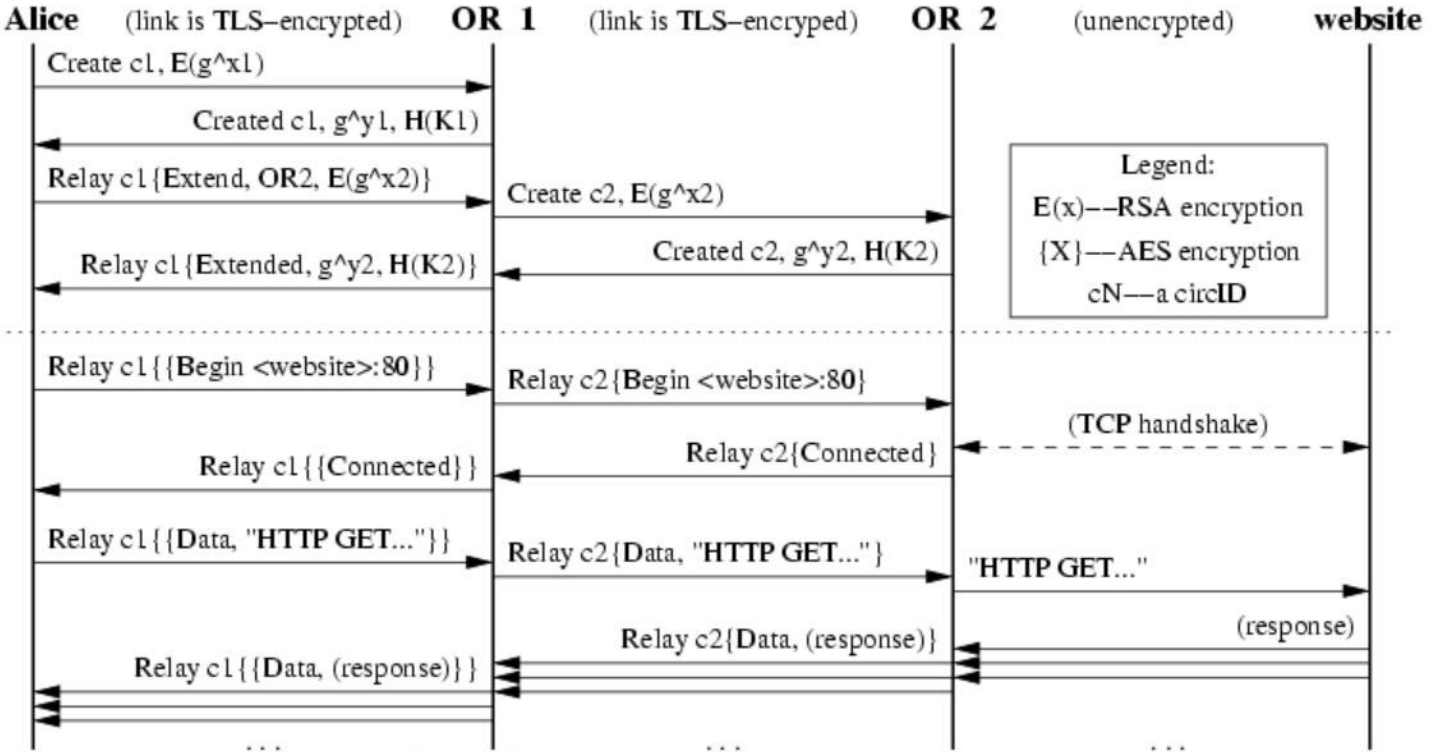
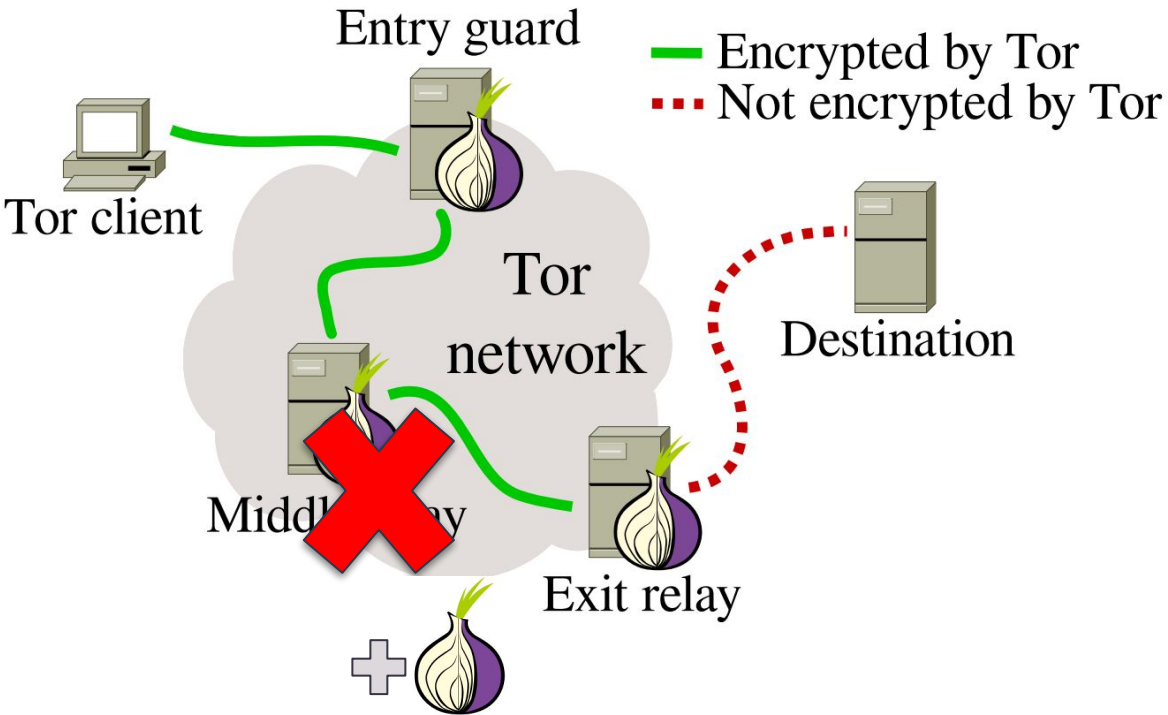



Figure 1: Alice builds a two-hop circuit and begins fetching a web page.

https://svn.torproject.org/svn/projects/design-paper/tor-design.html

# How many relay nodes?



# How many relay nodes?

A Tor Circuit Contains 3 Relays

1. Entry/Guard Node
2. Middle Node
3. Exit Node

Why not two?

If the attacker owns both relays, they can immediately sniff out who you are. With 3 relays, the middle relay can obscure what entry and exit node the attacker needs to own.

Why not more than 3 relays?

The problem comes down to cost, adding more relays does increase security in exchange for increased network load. However, you are still susceptible to traffic analysis if the attacker owns both the start and exit nodes. Therefore, it was determined that 3 relays was a good compromise.

https://www.torproject.org/docs/faq.html.en#ChoosePathLength

# Problems?

- What if there are hostile node owners (eg. government)?
- What if DDoS attacks are used?
- What if network analysis is used to establish patterns?
- Wouldn't adding each encryption layer add extra size to the message?
- The nodes in the circuit are chosen randomly.
- TOR handles load sharing. Rerouting failures to avoid detecting traffic paths.
- The route is reset by the TOR network every 10 minutes. Efficiency vs Security.
- "Cells" of fixed size.

# Some of TOR's Properties:*

- perfect forward secrecy
- Congestion control
- Directory servers
- Integrity checking
- configurable exit policies

# Perfect Forward Secrecy* ==

getting the key at some point in the future doesn't expose old messages

- By using Incremental or *telescoping* path-building design instead of a single multiply encrypted data structure

- By making Client negotiate session keys with each node in the circuit. Once these keys are deleted, subsequently compromised nodes cannot decrypt old traffic.

# What protocols are supported?*

- Tor uses the standard and near-ubiquitous SOCKS proxy interface so most TCP-based programs are supported without modification

# Many TCP streams can share one circuit*

- **Improved Efficiency**

  Avoid multiple public key operations for every request

- **Improved anonymity**

  Building so many circuits could be a threat to anonymity*

  This was excluded from the algorithm described in previous slides for simplicity

*https://www.freehaven.net/anonbib/cache/wright03.pdf

# Is any content filtering done?*

Using Privoxy:

A non-caching web proxy with filtering capabilities for enhancing privacy, manipulating cookies and modifying web page data and HTTP headers before the page is rendered by the browser”

- Web page filtering (text replacements, removes banners based on size, invisible "web-bugs" and HTML annoyances, etc.)
- Modularized configuration that allows for standard settings and user settings to reside in separate files, so that installing updated actions files won't overwrite individual user settings.
- Support for Perl Compatible Regular Expressions in the configuration files, and a more sophisticated and flexible configuration syntax.
- GIF de-animation.
- Bypass many click-tracking scripts (avoids script redirection).
- User-customizable HTML templates for most proxy-generated pages (e.g. "blocked" page).
- Auto-detection and re-reading of config file changes.

# Congestion Control*

- decentralized (no inter-node control communication or global views of traffic)

- uses end-to-end acks to maintain anonymity while allowing nodes at the edges of the network to detect congestion or flooding and send less data until the congestion subsides.

# Directory Servers*

- more trusted nodes act as *directory servers*
- provide signed directories describing known routers and their current state.
- Clients periodically download them via HTTP
- Better than the older approach of flooding status info to all nodes

# Online directory of TOR nodes

**Tor Network Status**

Tor Exit Node Query | TorStatus Server Details | Advanced Query Options | Advanced Display Options | Network Statistic Summary | Network Statistic Graphs

CSV List of Current Result Set | CSV List of All Current Tor Server IP Addresses | CSV List of All Current Tor Server Exit Node IP Addresses

List of AS by Router Count, sorted by router count | List of AS by Exit Router Count, sorted by router count | List of AS by Router Count, sorted by bandwidth | List of AS by Exit Router Count, sorted by bandwidth

You do not appear to be using Tor
Your IP Address is: 63.250.126.195

Legend:
Router is okay
Router is hibernating
Router is currently down
Router is a bad exit node

| Application Server Details | |
|---|---|
| Cache Last Updated (Local Server Time): | 2018-02-26 20:42:53 UTC |
| Last Update Cycle Processing Time (Seconds): | 517 |
| Number of Routers In Cache: | 6067 |
| Number of Descriptors In Cache: | 6962 |
| Approximate Page Generation Time (Seconds): | 0.1131 |

| Router Name | Bandwidth (KB/s) | Uptime | Hostname | ORPort | DirPort | Bad Exit | FirstSeen | ASName | ASNumber | ConsensusBandwidth KB/s |
|---|---|---|---|---|---|---|---|---|---|---|
| IPredator | 73555 | 9 d | exit1.ipredator.se [197.231.221.211] | 443 | 9030 | x | 2014-04-19 | CYBERDYNE, LR | 37560 | 253000 |
| novatorrelay | 53323 | 53 d | no-reverse-dns-configured.com [93.174.93.71] | 443 | 9030 | x | 2017-09-05 | QUASINETWORKS, NL | 29073 | 189000 |
| spechttor1 | 46209 | 42 d | chili.kuehrmann.net [138.201.169.12] | 443 | 80 | x | 2016-12-16 | HETZNER-AS, DE | 24940 | 78000 |
| 0x3d004 | 44486 | 25 d | snowden.pep-security.net [62.138.7.171] | 9001 | 9030 | x | 2016-08-24 | GD-EMEA-DC-SXB1, DE | 8972 | 85700 |
| 0x3d005 | 43226 | 25 d | snowden.pep-security.net [62.138.7.171] | 8001 | 8030 | x | 2016-09-23 | GD-EMEA-DC-SXB1, DE | 8972 | 99800 |
| poiuty4 | 42352 | 11 d | ns3083447.ip-145-239-6.eu [145.239.6.131] | 9001 | 9030 | x | 2017-12-22 | OVH, FR | 16276 | 104000 |
| fastprivacyo1 | 41293 | 8 d | 142.51-175-193.customer.lyse.net [51.175.193.142] | 443 | 80 | x | 2017-06-11 | ALTIBOX_AS Norway, NO | 29695 | 90600 |
| Multivac | 41100 | 6 d | multivac.io [163.172.53.84] | 21 | 143 | x | 2014-04-08 | AS12876, FR | 12876 | 112000 |
| poiuty2 | 38987 | 11 d | ns542112.ip-144-217-255.net [144.217.255.69] | 9001 | 9030 | x | 2017-08-29 | OVH, FR | 16276 | 115000 |
| quadhead | 32699 | 13 d | tor3.quadhead.de [148.251.190.229] | 9010 | 9030 | x | 2015-01-19 | HETZNER-AS, DE | 24940 | 13700 |
| DipulseIT2 | 32192 | 3 d | 62-210-82-83.rev.poneytelecom.eu [62.210.82.83] | 8080 | None | x | 2018-01-24 | AS12876, FR | 12876 | 93200 |
| TokenBlack | 31996 | 11 d | freebird.system33.pw [144.217.254.208] | 9001 | 9002 | x | 2017-05-30 | OVH, FR | 16276 | 148000 |
| poiuty1 | 31632 | 11 d | ns3060920.ip-5-39-64.eu [5.39.64.7] | 9001 | 9030 | x | 2016-01-26 | OVH, FR | 16276 | 26300 |
| atlantis | 30359 | 48 d | atlantic746.serverprofi24.com [85.25.43.31] | 443 | 80 | x | 2017-10-27 | GD-EMEA-DC-SXB1, DE | 8972 | 105000 |
| Despoina | 29541 | 18 d | h-35-25.A357.priv.bahnhof.se [94.254.35.25] | 6881 | 6882 | x | 2014-04-09 | BAHNHOF http://www.bahnhof.net/, SE | 8473 | 75400 |
| TotorBE2 | 29509 | 12 d | ip178.ip-5-39-33.eu [5.39.33.178] | 9001 | 9030 | x | 2016-12-18 | OVH, FR | 16276 | 55100 |
| 0x3d002 | 29424 | 25 d | 0x3d.lu [91.121.23.100] | 9001 | 9030 | x | 2014-04-22 | OVH, FR | 16276 | 57100 |
| TotorBE1 | 28229 | 7 d | ip176.ip-5-39-33.eu [5.39.33.176] | 9001 | 9030 | x | 2016-10-22 | OVH, FR | 16276 | 69000 |
| apx2 | 27915 | 6 d | tor-exit.r2.apx.pub [185.38.14.171] | 9001 | 9030 | x | 2015-02-02 | YISP-AS, NL | 58073 | 79000 |
| CryoBBNx | 26829 | 6 d | ip-51-254-45-43.ddhosts.net [51.254.45.43] | 9001 | None | x | 2017-07-25 | OVH, FR | 16276 | 52300 |
| proxydotshop | 26668 | 23 d | ns3031342.ip-149-202-94.eu [149.202.94.25] | 9001 | 9030 | x | 2018-02-03 | OVH, FR | 16276 | 103000 |
| Onyx | 26557 | 40 d | onyx.ip-eend.nl [192.42.115.102] | 9004 | 80 | x | 2015-04-22 | SURFNET-NL SURFnet, The Netherlands, NL | 1103 | 73900 |
| Chenjesu | 26312 | 25 d | ip-54-36-205-38.ddhosts.net [54.36.205.38] | 9001 | None | x | 2017-10-02 | OVH, FR | 16276 | 67400 |
| bengalfox | 26291 | 45 d | bengalfox.doridian.net [62.210.251.189] | 9001 | 9030 | x | 2017-12-14 | AS12876, FR | 12876 | 57200 |
| xanaduregio | 26125 | 14 d | xanaduregio.emeraldonion.org [23.129.64.102] | 443 | 80 | x | 2017-08-30 | EMERALD-ONION - Emerald Onion, US | 396507 | 37900 |
| cry | 25994 | 40 d | cry.ip-eend.nl [192.42.115.101] | 9003 | 8080 | x | 2015-04-22 | SURFNET-NL SURFnet, The Netherlands, NL | 1103 | 31800 |
| bonjour1 | 25885 | 3 d | loft9169.serverprofi24.com [188.138.33.233] | 443 | 80 | x | 2017-10-27 | GD-EMEA-DC-SXB1, DE | 8972 | 95400 |
| Kimchi | 25573 | 49 d | ns3091493.ip-54-36-120.eu [54.36.120.156] | 443 | 80 | x | 2017-11-23 | OVH, FR | 16276 | 111000 |
| WreckingBytes | 25511 | 28 d | 178.132.4.123 [178.132.4.123] | 443 | 80 | x | 2017-12-09 | WORLDSTREAM, NL | 49981 | 13200 |
| TORion | 25482 | 67 d | ns500599.ip-192-99-34.net [192.99.34.48] | 443 | 80 | x | 2017-10-24 | OVH, FR | 16276 | 83100 |
| apx1 | 25411 | 6 d | tor-exit.r1.apx.pub [185.38.14.215] | 9001 | 9030 | x | 2014-11-04 | YISP-AS, NL | 58073 | 49300 |
| 0x3d001 | 25168 | 25 d | 0x3d.lu [91.121.23.100] | 8001 | 8030 | x | 2016-05-31 | OVH, FR | 16276 | 56800 |
| TORsitan | 24959 | 64 d | 172.241.140.26 [172.241.140.26] | 443 | 80 | x | 2016-08-24 | LEASEWEB-USA-NYC-11 - Leaseweb USA, Inc., US | 396362 | 39500 |
| par1 | 24833 | 2 d | 163-172-110-128.rev.poneytelecom.eu [163.172.110.128] | 9001 | 9030 | x | 2018-02-05 | AS12876, FR | 12876 | 118000 |
| ButtersStotch | 24109 | 6 d | thisis.feralhosting.com [185.21.216.198] | 9001 | 9002 | x | 2017-04-16 | FERAL Feral Hosting, GB | 200052 | 63900 |

http://torstatus.blutmagie.de/

# Integrity Checking*

**Why is integrity checking critical?**

Any node on the circuit could change the contents of data cells as they passed by — for example, to alter a connection request so it would connect to a different webserver, or to 'tag' encrypted traffic and look for corresponding corrupted traffic at the network edges!

# Configurable Exit Policies*

- Each node advertises a policy describing the hosts and ports to which it will connect

- Safer for exit node volunteers because they can control the types of traffic that will exit from their nodes

# Ways of using TOR

- TOR browser
  - Uses NoScript
  - “HTTPS everywhere” Firefox extension
  - Using other browsers is dangerous
- Other products
- Write your own program using the related API: https://stem.torproject.org

# Moving on to …

# The Deep, Dark, Web

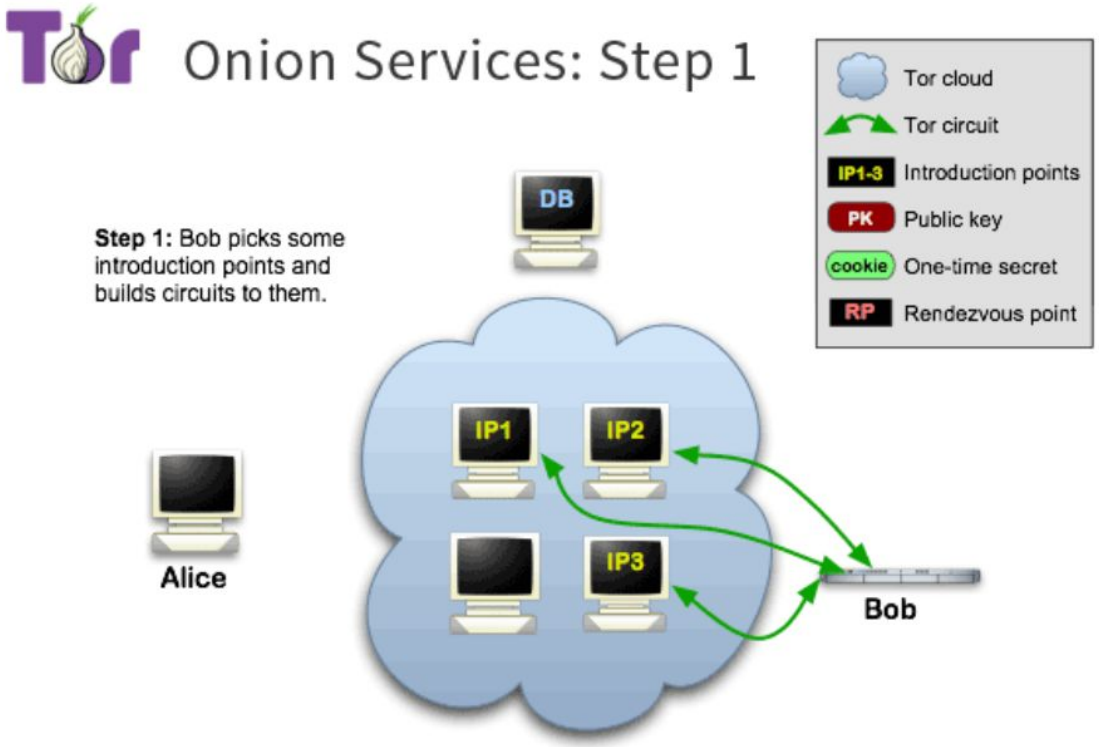
The **Surface Web** is anything indexed by a search engine like Google, Yahoo, Bing, etc. As of 2016 Google indexes over 130 trillion pages, with only 1 trillion pages indexed in 2008.

The **Deep Web** is anything that isn't indexed by a search engine like Google, Yahoo, Bing, etc. This includes dynamic pages that can only be viewed with special permissions, cookies, cache, etc. It's estimated the Deep Web is 500x bigger than the Surface web.

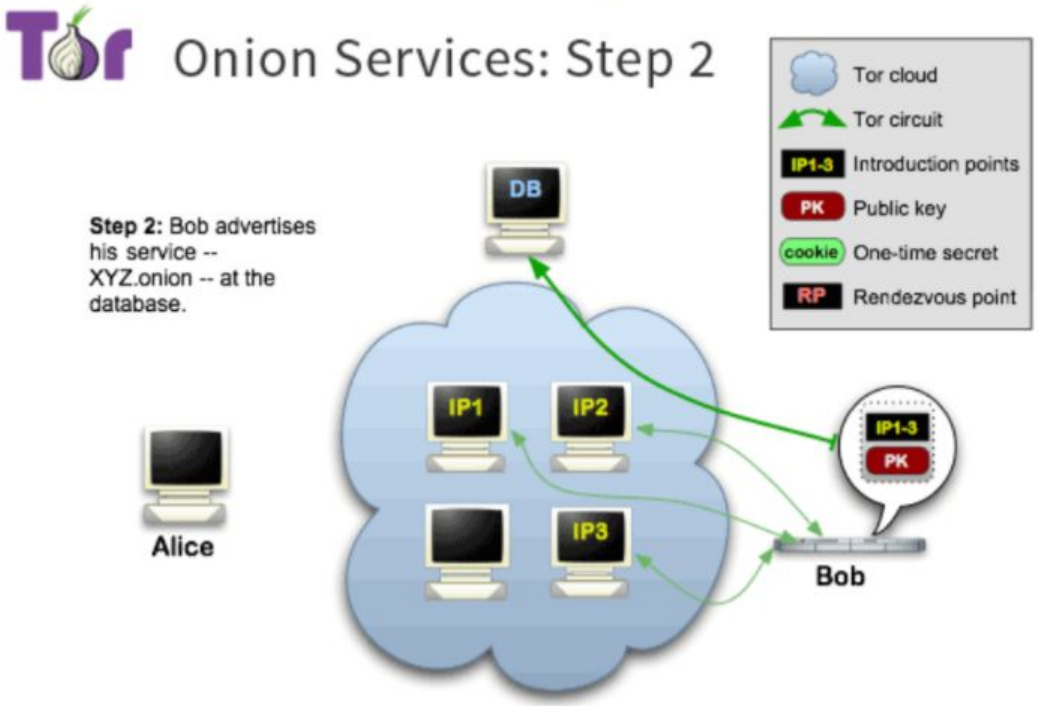
The **Dark Web** is a subset of the Deep Web. The unique feature of pages in the Dark Web is that they follow a unique protocol to encrypt information differently, that is standard browsers cannot read and decrypt these sites.

These protocols include TOR's onion service, I2P's standard protocol, and etc. These protocols helps preserve the service hosts anonymity while making themselves known to the network.

# Hidden Service(Rendezvous) Protocol



# Hidden Service Protocol

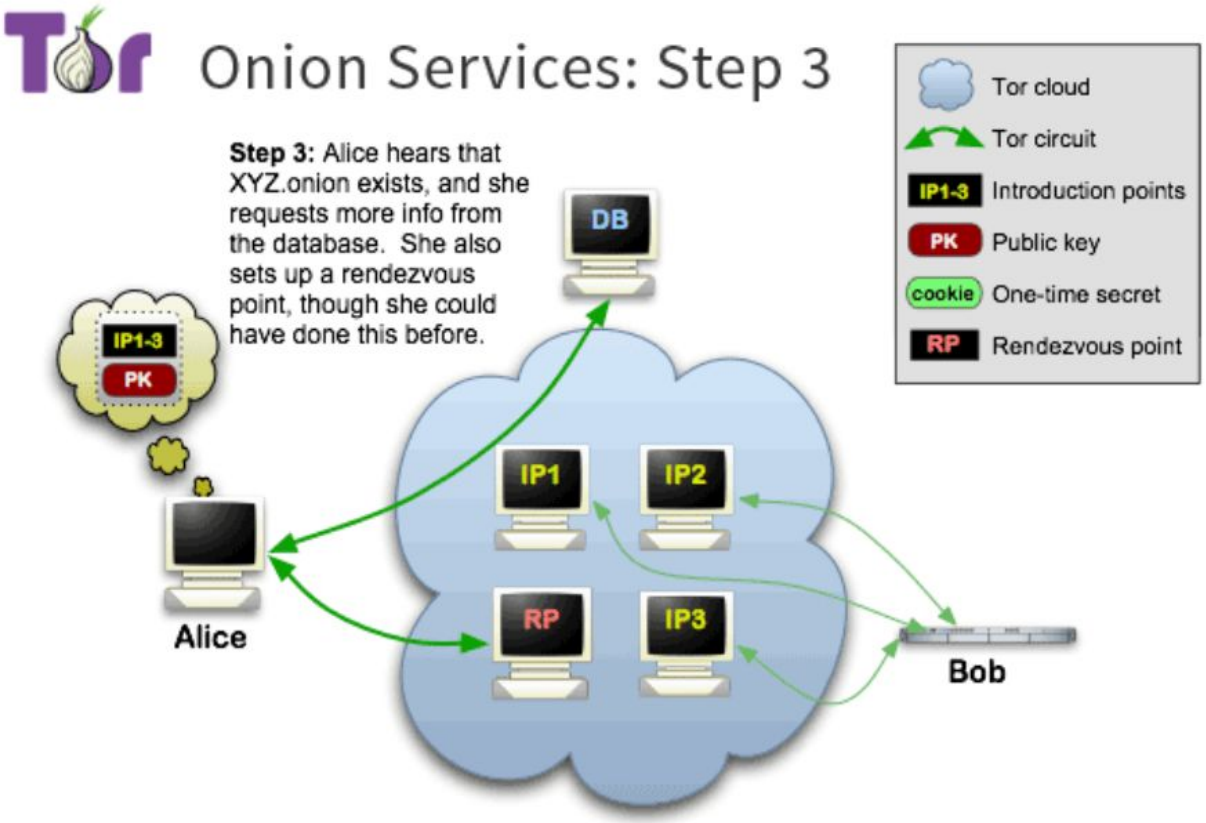



The Hidden Service calculates it's key pair

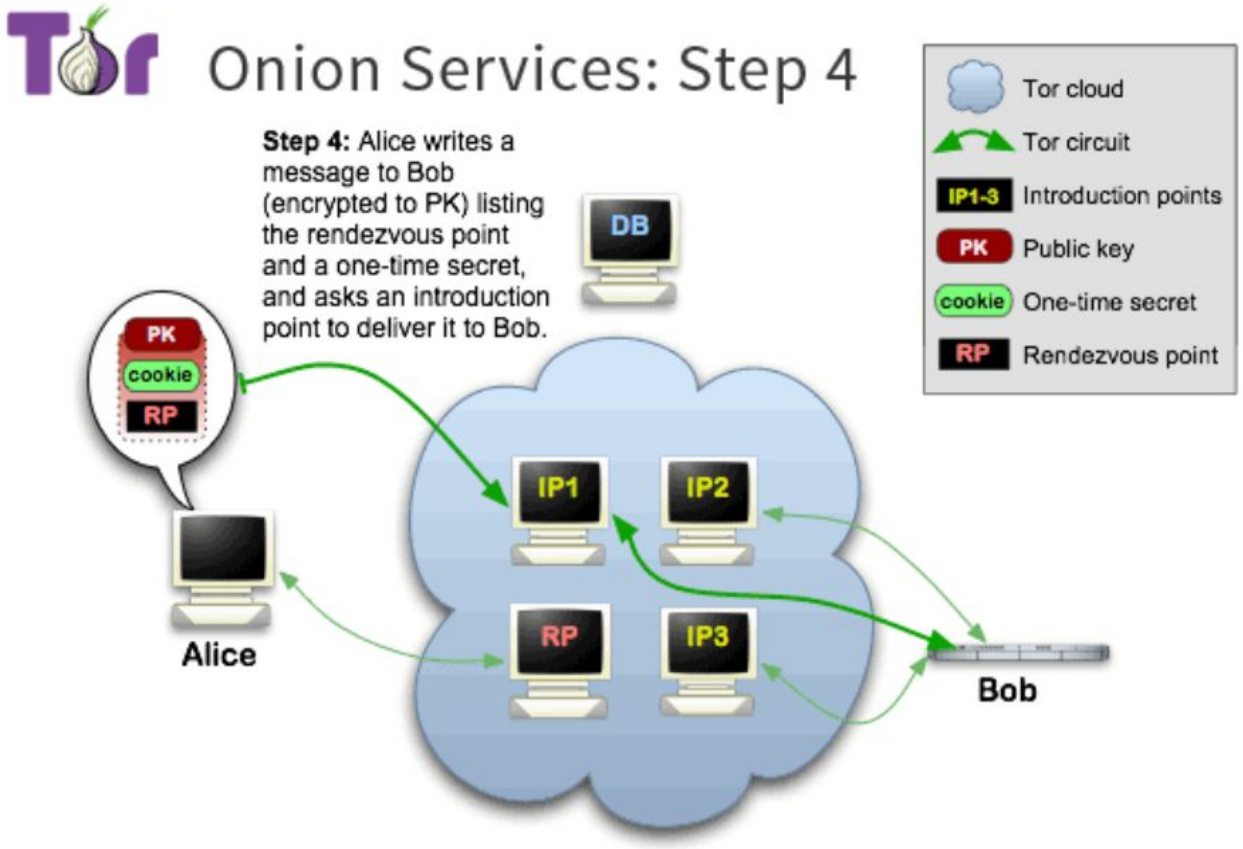
The hidden service creates a Service Descriptor containing it's Public key for Authentication and the IP Addresses of the Relays acting as introduction points. The Service Descriptor gets signed with the hosts private key.

The Service Descriptor now gets added to a Distributed Hash Table, where each relay holds a block of the table.

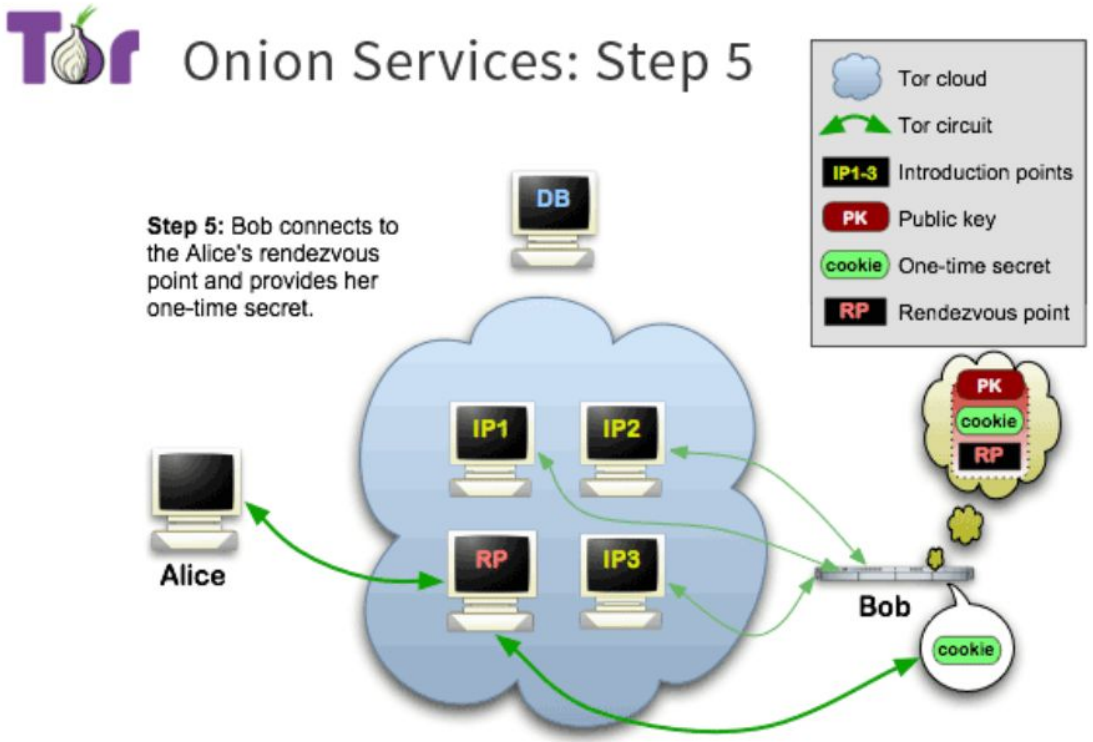
# Hidden Service Protocol



# Hidden Service Protocol

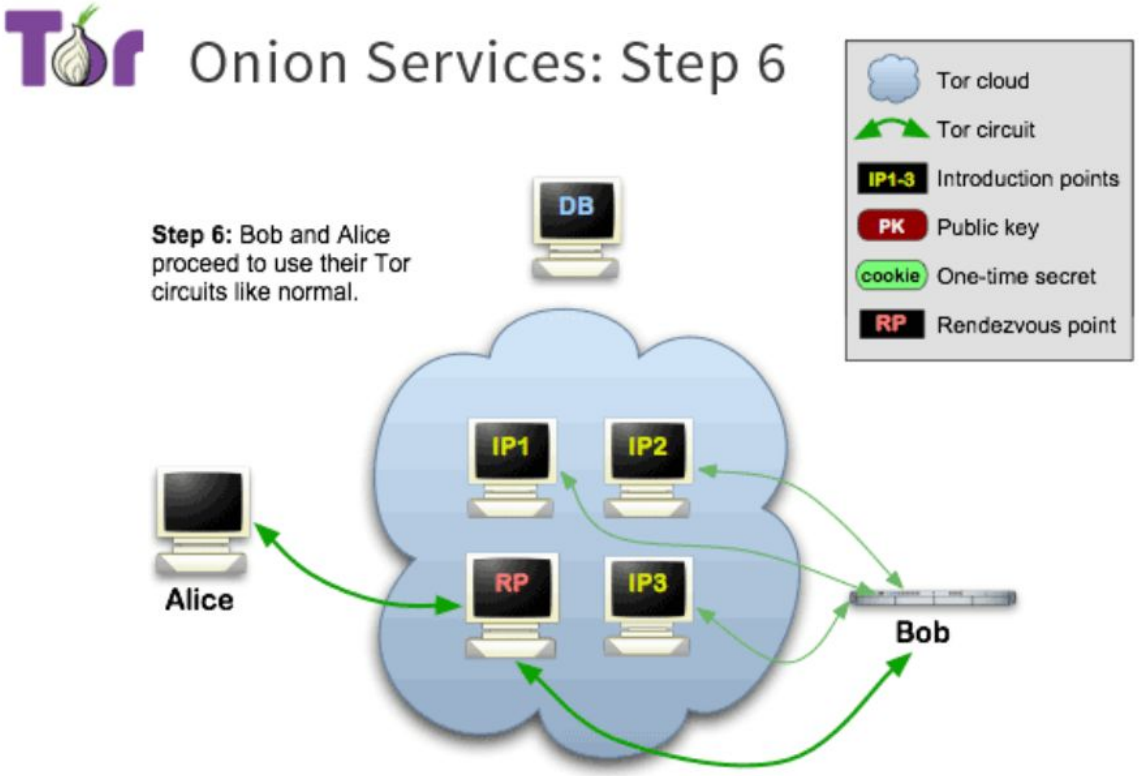


# Hidden Service Protocol




The hidden service decrypts the clients encrypted message with their private key which contains the rendezvous point and secret message. The service now creates a circuit to the rendezvous point if they accept the message and sends a one-time secret to the rendezvous point. The rendezvous point now establishes the connection between the client and the hidden service.

# Hidden Service Protocol



# De-Anonymization

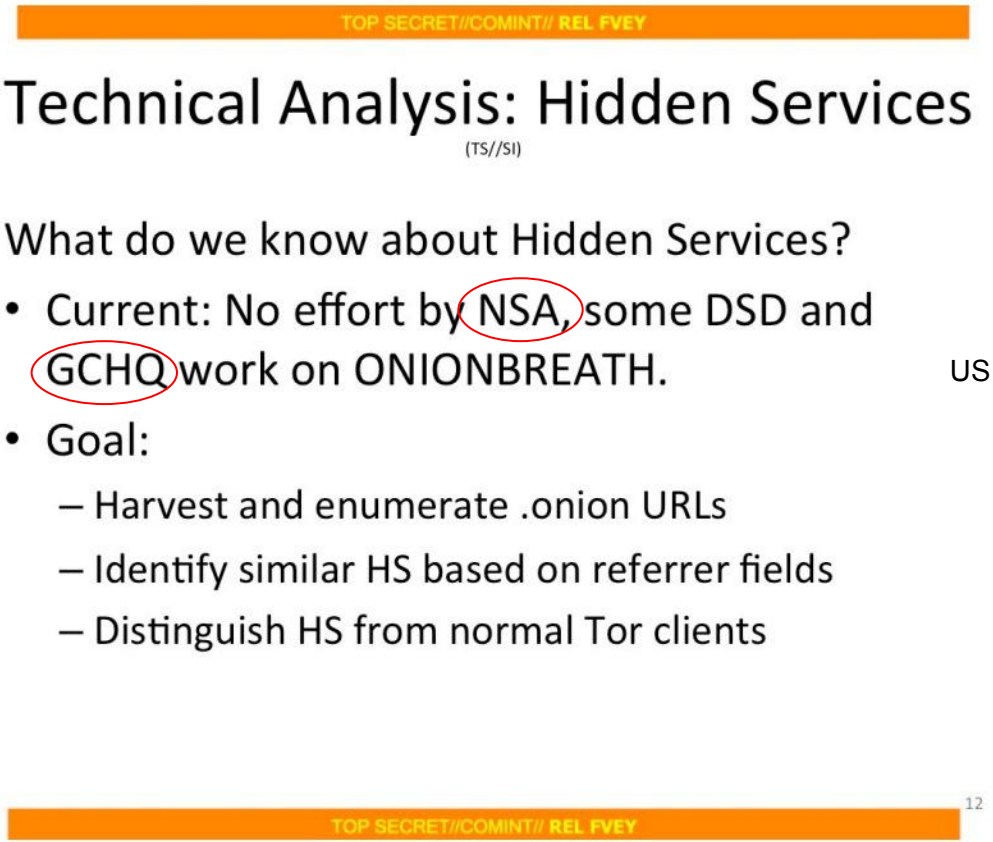
Traffic Analysis:

- Timing Analysis
- Circuit reconstruction using hostile nodes (limited success)
- Dumb users (EPICFAIL)

Sensitive Information:

Websites can tell if traffic comes from a relay, sensitive information can give out client identity. Un-Encrypted Data containing sensitive information can also give away identity to exit node.

# NSA internal document




UK intelligence

US Intelligence

https://edwardsnowden.com/docs/doc/tor-stinks-presentation.pdf

# Attack: Javascript Exploit(2013)

Attack:

Malicious JavaScript exploited a TOR/Firefox memory-management vulnerability, forcing users to send out a unique identifier to a server. Exploit also contained "heap spraying" used to bypass security protection and detection.

TOR's Response:

Since the alleged vulnerabilities and exploits commonly related to JavaScript and Flash. TOR has since been packaged with NoScript; disabling most add-ons and plugins such as Javascript, Java, Flash, etc.

Prerequisites: Requires attacker to be the host of the hidden service, which executes the exploit.

# Additional Info : CVE-2013-1690

Context:

An issue with specially crafted web content using the "onreadystatechange" event may cause a crash in which unmapped memory is executed.

Use:

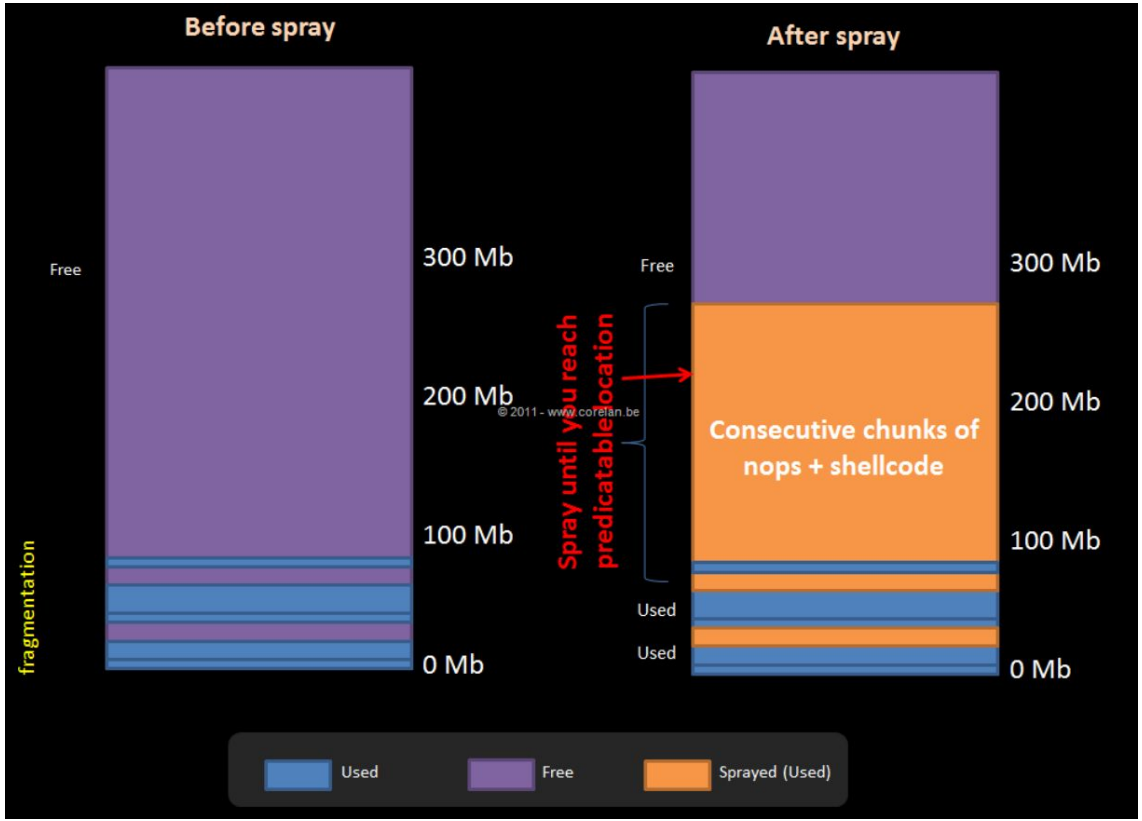
It was suspected that the FBI used this exploit to deliver a heap sprayed payload.

Vulnerability & Exploit Database:

https://blog.rapid7.com/2013/08/07/heres-that-fbi-firefox-exploit-for-you-cve-2013-1690/

https://www.rapid7.com/db/modules/exploit/windows/browser/mozilla_firefox_onreadystatechange

# Additional Info: Heap Spraying




Heap Spraying is used to hide a payload inside a chunk of data. By itself it is not a security issue, however given a separate security issue like the firefox memory-management vulnerability, heap spraying allows an attacker to deliver the payload undetected.

# Attack: Packet Analyzing(2015)

Attack:

Gather network data on a pre-determined list of hidden services in advance. Analyze patterns in number of packets being passed between the hidden service and entry relay allowed researchers to obtain a unique fingerprint for the service. With machine learning, researchers were able to pinpoint the host with 88% certainty.

Problems:

-Pattern analyzation fails when you pad the traffic.

-2.9% False Positive rate is enormous when thousands or millions of pages are being browsed every second. Packet analyzing becomes too expensive.

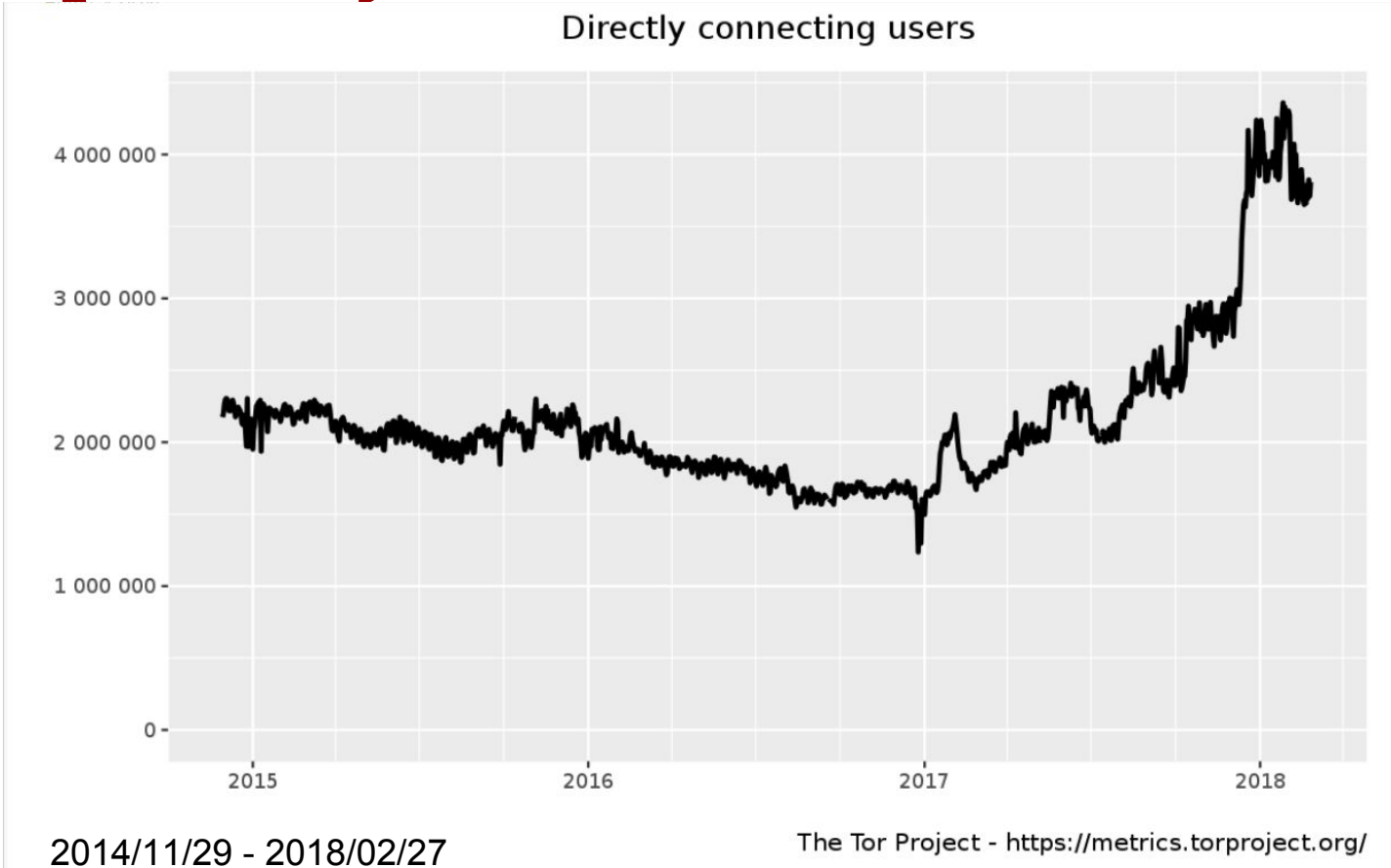
https://arstechnica.com/information-technology/2015/07/new-attack-on-tor-can-deanonymize-hidden-services-with-surprising-accuracy/

# Detecting and blocking TOR

- Hard since it looks just like https. Uses same ports. Could use statistical to distinguish different SSL protocols.

- Block all connections to known TOR relays. (Solution: TOR bridges)
- MediaWiki TorBlock extension automatically restricts edits made through Tor

# Popularity


Directly connecting users
4 000 000
3 000 000
2 000 000
1 000 000
0
2015
2016
2017
2018
2014/11/29 - 2018/02/27
The Tor Project - https://metrics.torproject.org/

# Sources


- https://css.csail.mit.edu/6.858/2011/lec/l17-tor.txt
- https://edwardsnowden.com/docs/doc/tor-stinks-presentation.pdf [leaked NSA internal doc]
- https://media.torproject.org/video/2011-12-28-28c3-4800-en-how_governments_have_tried_to_block_tor_h264.mp4 [from the makers of Tor]
- https://svn.torproject.org/svn/projects/presentations/slides-28c3.pdf
- https://www.youtube.com/watch?v=LAcGiLL4OZU
- https://www.torproject.org/docs/documentation.html.en
- https://arstechnica.com/information-technology/2013/08/attackers-wield-firefox-exploit-to-uncloak-anonymous-tor-users/
- https://arstechnica.com/information-technology/2015/07/new-attack-on-tor-can-deanonymize-hidden-services-with-surprising-accuracy/
- https://en.wikipedia.org/wiki/Heap_spraying
- https://www.torproject.org/docs/onion-services.html.en
- https://www.torproject.org/docs/faq.html.en#ChoosePathLength
- https://searchengineland.com/googles-search-indexes-hits-130-trillion-pages-documents-263378
- https://www.youtube.com/watch?v=oiR2mvep_nQ
- https://www.youtube.com/watch?v=joxQ_XbsPVw


# Sources. Continued.


- https://www.youtube.com/watch?v=QRYzre4bf7I
- https://www.icann.org/news/blog/the-dark-web-the-land-of-hidden-services
- https://www.torproject.org/download/download-easy.html.en#warning
- https://www.youtube.com/watch?v=lVcbq_a5N9I