

Dissect Tor Bridge and Pluggable Transport



Xiaopeng Zhang & Peixue Li

Fortinet's FortiGuard Labs

Who We Are?

- **Xiaopeng Zhang**
 - ✓ Senior security researcher at Fortinet's FortiGuard Labs
 - ✓ Has worked in cyber security for more than 13 years
 - ✓ Email: xpzhang@fortinet.com

- **Peixue Li**
 - ✓ Director at Fortinet's FortiGuard Labs
 - ✓ Has worked in cyber security for more than 15 years
 - ✓ Email: pxli@fortinet.com

Why We Did This Research?

- **Some customers need to identify Tor traffic**
- **Evaluate the security of Tor network**
- **Monitor the threats in dark web**

Agenda

- **Introduction**
- **The Tor Network**
- **Anti-Censorship**
 - **The Built-In Obfs4 Bridges**
 - **How Tor Client Connects To Obfs4 Bridge**
 - **How Obfs4 Transforms Tor-Encrypted Traffic**
- **Conclusion**
- **Q&A**

Introduction

What Is TOR?

- An open source project for **anonymous communication** and the name is derived from its original project name called “**The Onion Router**”
- Tor traffic goes through a worldwide overlay network comprising thousands of volunteer-run relays to **conceal users’ identity, location and online activity** from network surveillance or traffic analysis (client side anonymity)
- Tor client periodically creates **virtual circuits** comprising 3 randomly-selected relays through the Tor network, then routes traffic to the destination using **onion routing technique**

What Is TOR? (continued)

- Tor network also provides **anonymous onion service** (e.g. websites) which can host censorship-resistant content (server side anonymity)
- An onion service is accessed through its **onion address** usually via the Tor browser
- **Tor browser** (<https://www.torproject.org>) is built based on Mozilla Firefox
- Tor also provides features for **anti-censorship**

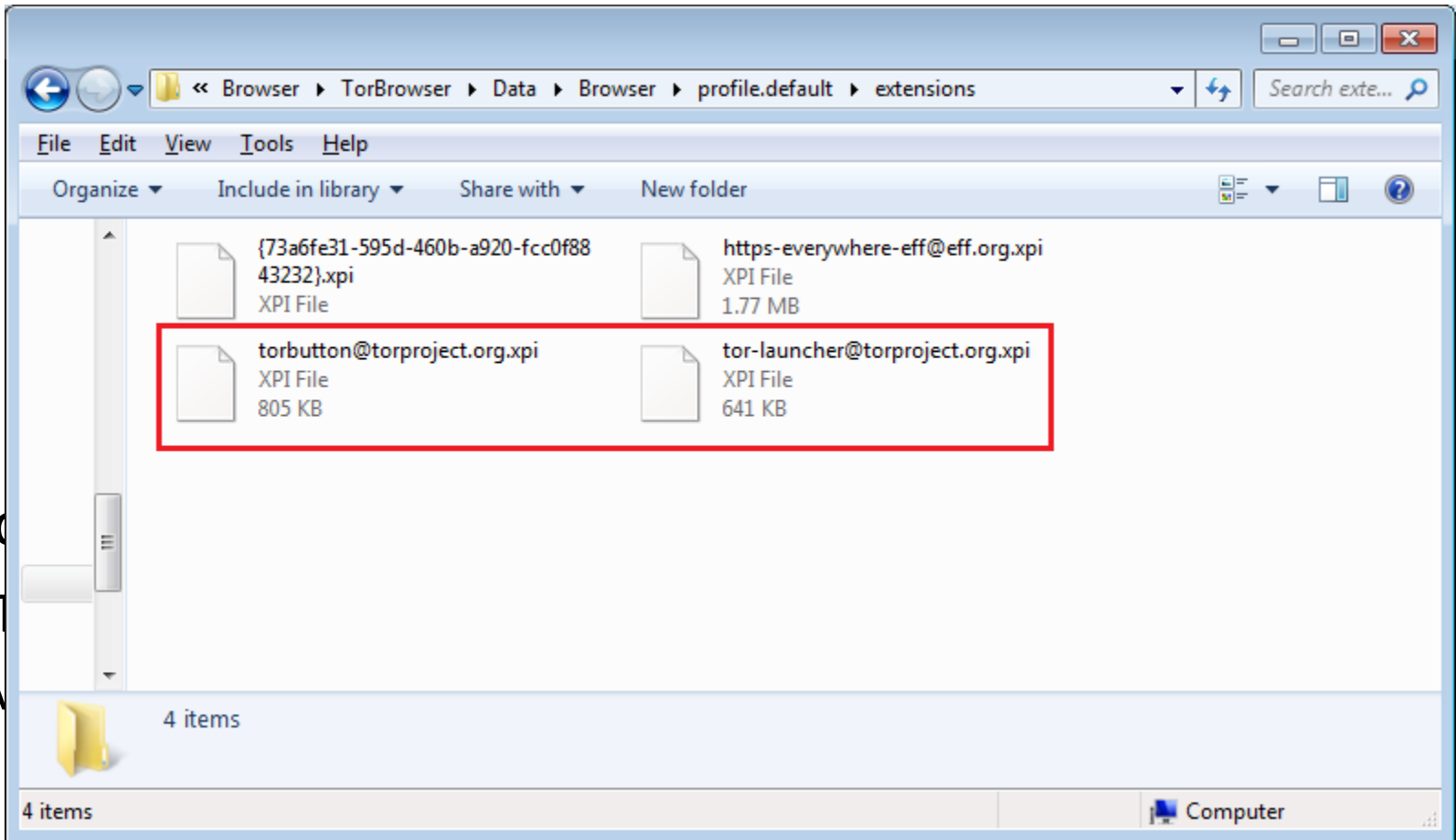
Two Firefox Extensions For Tor

- **TorLauncher** is in charge of starting Tor main process tor.exe.
- **Torbutton** manages all interfaces about Tor, such as Tor Network Settings, Tor Circuit, Tor About and so on.

Location:

`"TOR_INSTALLATION_FOLDER\Browser\TorBrowser\Data\Browser\profile.default\extensions\"`

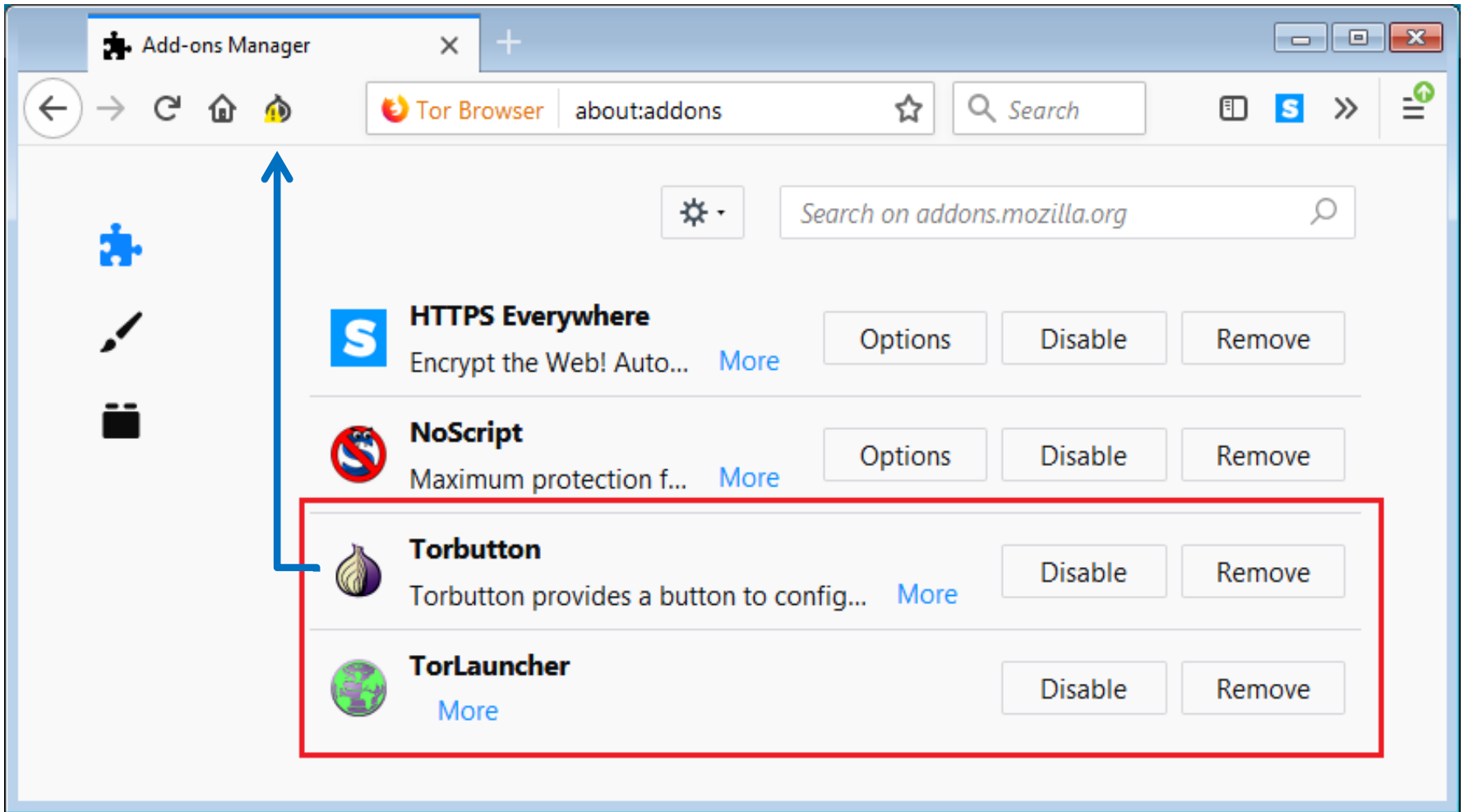
Two Firefox Extensions For Tor



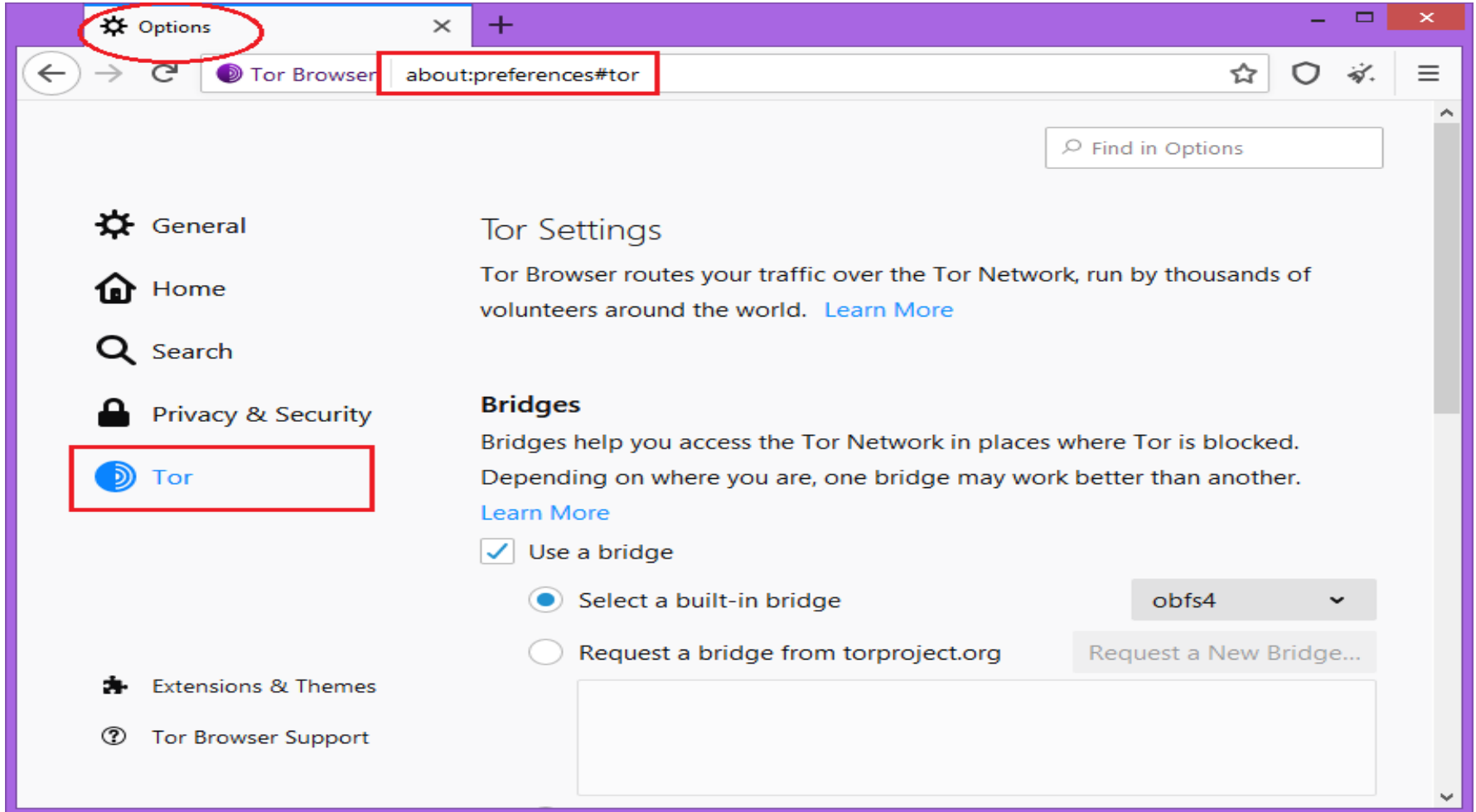
Lo
"T
r\

se

Two Firefox Extensions For Tor

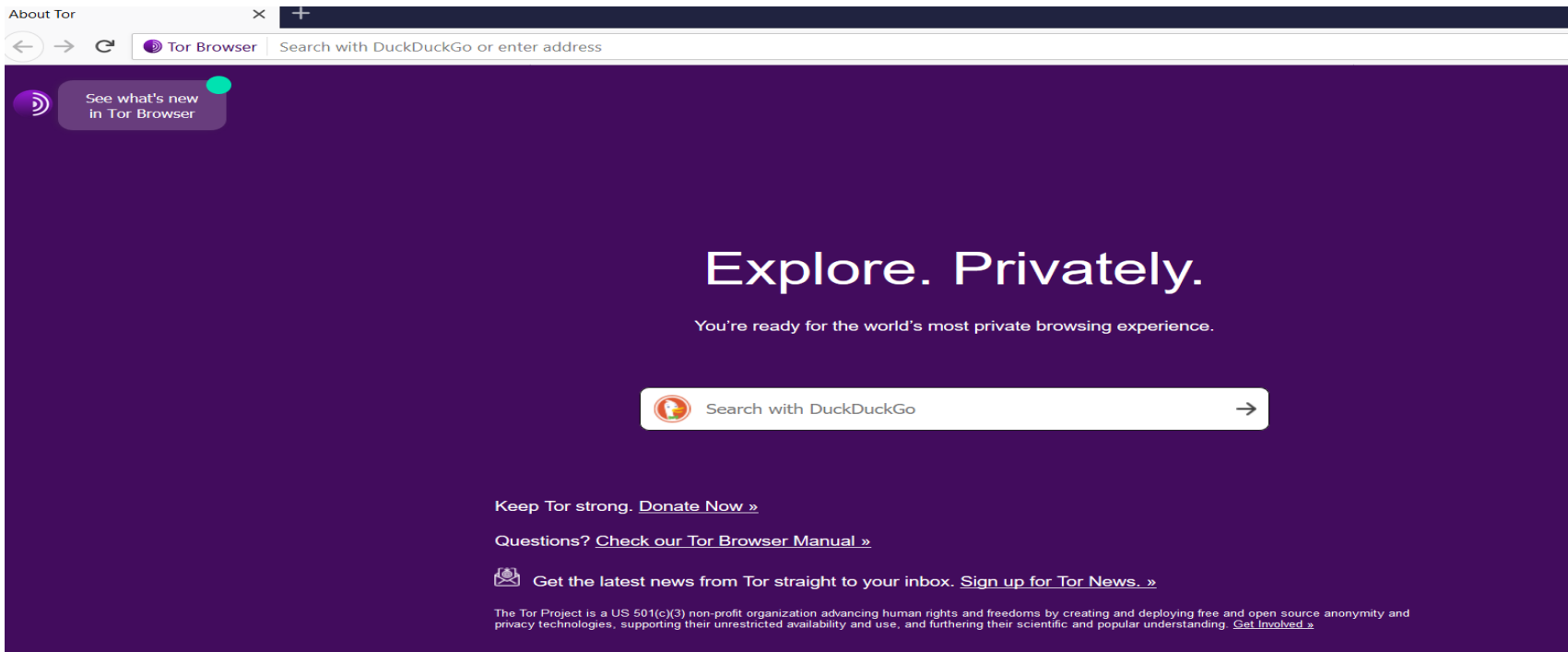


Extensions Integrated Into Options Since Tor Browser 9.0



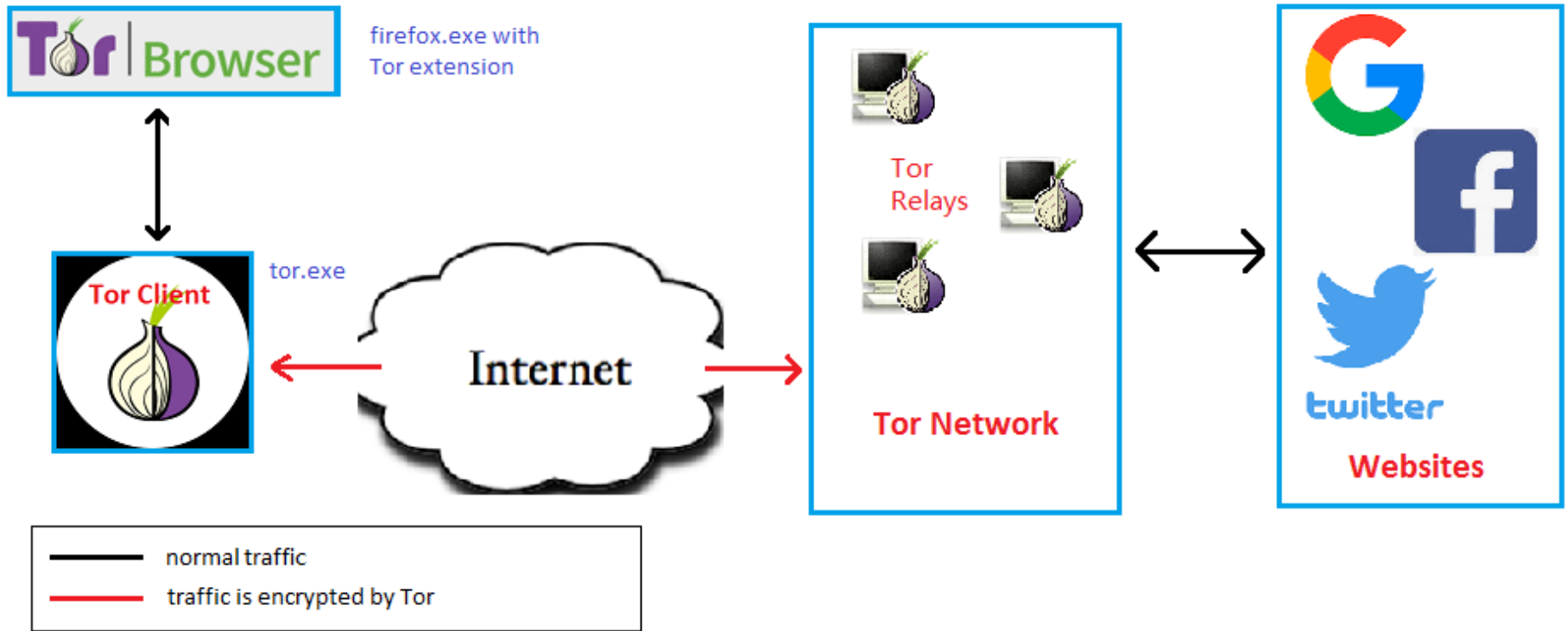
Analysis Environment

- **Windows 7 32-bit SP1**
- **Tor Browser 8.0** (based on Firefox 60.2.0esr)
- **TorLauncher 0.2.16.3** (extension)
- **Torbutton 2.0.6** (extension)



The Tor Network

Tor Communication Flow Chart



Tor Relays

- Most Tor relays are listed in the **main Tor directory** which can be accessed by anyone
- **Tor network status** can be found on <http://torstatus.blutmagie.de/>

Router Name	Bandwidth (KB/s)	Uptime	Hostname	ORPort	DirPort	Bad Exit	FirstSeen	ASName
drjohn	58986	270 d	ns3129544.ip-51-75-144.eu [51.75.144.67]	443	8080	✗	2018-12-21	OVH, FR
ExitNinja	57850	15 d	46.165.245.154 [46.165.245.154]	443	80	✗	2014-11-06	LEASEWEB-DE-FRA-10, DE
Unnamed	57850	211 d	ns3082025.ip-145-239-66.eu [145.239.66.236]	9001	9030	✗	2018-12-12	OVH, FR
BigBen	57220	248 d	95.154.221.4 [95.154.221.4]	9001	9030	✗	2019-01-11	IOMART-AS, GB
faoslegion	55784	24 d	92.62.139.103 [92.62.139.103]	9001	9030	✗	2019-07-19	BALTNETA Customers AS, LT
Unnamed	54934	280 d	ns3128209.ip-51-68-206.eu [51.68.206.35]	9001	9030	✗	2018-12-11	OVH, FR
SergeiK	54660	250 d	95.154.221.3 [95.154.221.3]	9001	9030	✗	2019-01-10	IOMART-AS, GB
Unnamed	53504	280 d	static.57.151.46.78.clients.your-server.de [78.46.151.57]	9001	9030	✗	2018-12-11	HETZNER-AS, DE
Unnamed	53130	273 d	78.129.150.83 [78.129.150.83]	9001	9030	✗	2018-12-18	IOMART-AS, GB
sheldon	52248	248 d	95.154.221.6 [95.154.221.6]	9001	9030	✗	2019-01-11	IOMART-AS, GB
privacyguardian	51761	274 d	ns3128207.ip-51-68-206.eu [51.68.206.28]	9001	9030	✗	2018-12-17	OVH, FR
UbuntuTor	51688	249 d	95.154.221.2 [95.154.221.2]	9001	9030	✗	2019-01-10	IOMART-AS, GB
torrelay01	51043	273 d	78.129.150.54 [78.129.150.54]	9001	9030	✗	2018-12-18	IOMART-AS, GB
LittleFoxRahja	50758	195 d	rahja.lf-net.org [178.63.72.24]	9001	9030	✗	2019-03-05	HETZNER-AS, DE
Unnamed	49825	274 d	ns3102095.ip-145-239-255.eu [145.239.255.86]	9001	9030	✗	2018-12-17	OVH, FR
Unnamed	49325	110 d	ns3066555.ip-176-31-229.eu [176.31.229.76]	9001	9030	✗	2018-12-11	OVH, FR
drakeforce1	49122	277 d	ns3082988.ip-145-239-6.eu [145.239.6.189]	9001	9030	✗	2018-12-13	OVH, FR
Nordiques	49021	22 d	ns542132.ip-144-217-255.net [144.217.255.89]	9001	9030	✗	2019-08-22	OVH, FR
martinsrelay	48938	274 d	ns3104827.ip-54-36-164.eu [54.36.164.176]	9001	9030	✗	2018-12-17	OVH, FR
Unnamed	48159	272 d	static.213-239-204-62.clients.your-server.de [213.239.204.62]	9001	9030	✗	2018-12-19	HETZNER-AS, DE
PIAzhexit	46963	23 h	zrh-exit.privateinternetaccess.com [195.206.105.217]	443	80	✗	2018-12-21	M247, GB
hyacinthinus	46781	166 d	94.23.150.81 [94.23.150.81]	443	80	✗	2017-05-01	OVH, FR
Unnamed	46575	279 d	static.205.69.243.136.clients.your-server.de [136.243.69.205]	9001	9030	✗	2018-12-12	HETZNER-AS, DE
5efd8dd	46375	14 d	5efd8dd.retik.eu [213.239.216.221]	443	None	✗	2019-03-06	HETZNER-AS, DE

Access Website Through Tor Network

To access a destination through Tor network, a **virtual circuit** should be created first.

The screenshot shows a web browser window with the address bar displaying `https://www.google.com/#cns=0`. The page content shows the Google logo and search bar. A sidebar on the left displays the Tor circuit details for the current site, which is highlighted with a red border. The circuit path is as follows:

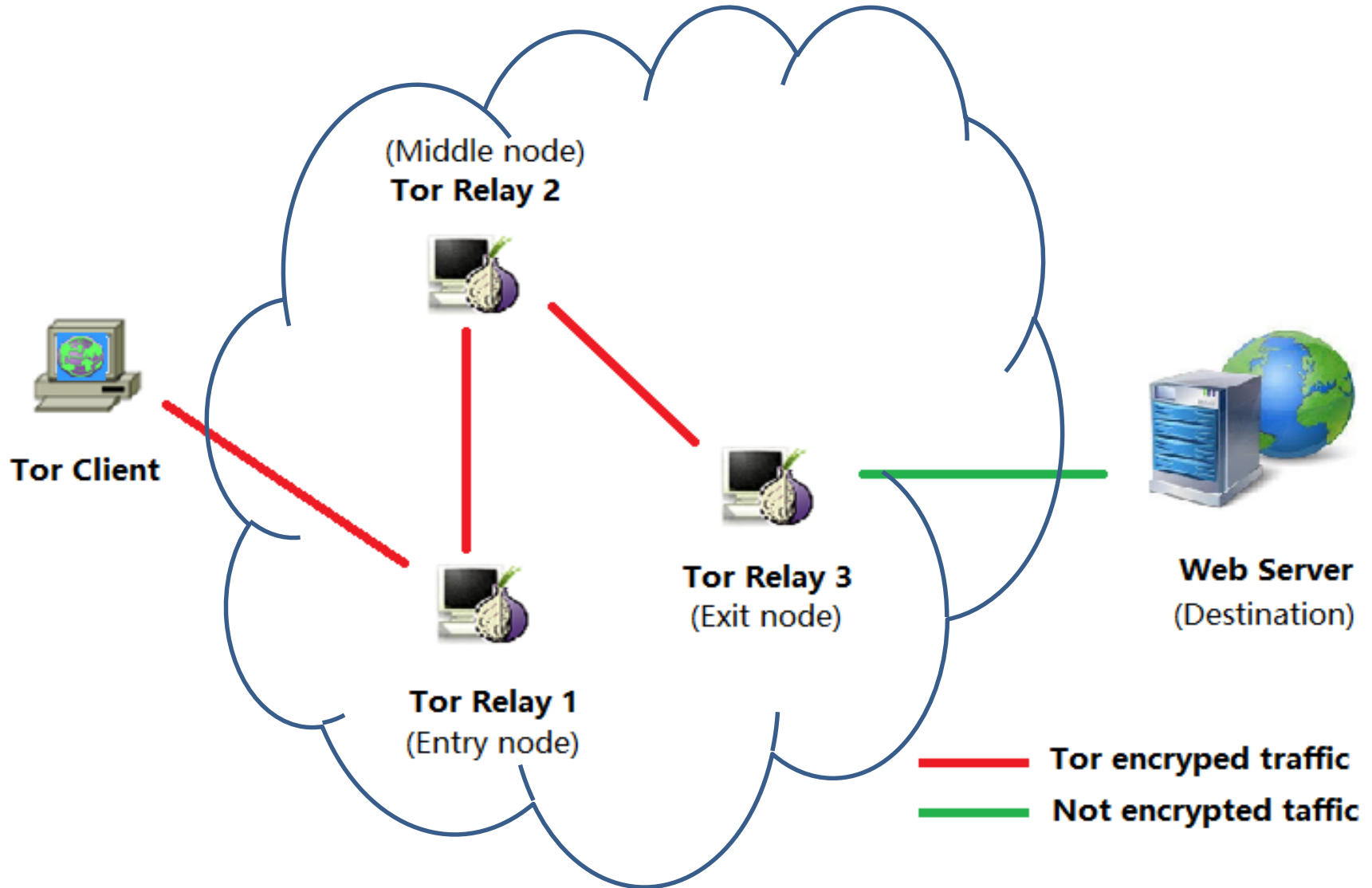
- Tor Circuit
- This browser
- United Kingdom 178.18.122.109 **Guard**
- Germany 136.243.131.29
- Netherlands 77.247.181.166
- google.com

Below the circuit details, there is a blue button labeled "New Circuit for this Site" and a note: "Your **Guard** node may not change. [Learn more](#)".

At the bottom of the sidebar, there is a "Permissions" section with a checkmark icon and the text: "You have not granted this site any special permissions."

At the bottom of the browser window, there are two buttons: "Google zoeken" and "Ik doe een gok". Below these buttons, it says "Google aangeboden in: [Frysk](#) [English](#)".

Tor Network & Circuit



How Tor Circuit is Created?

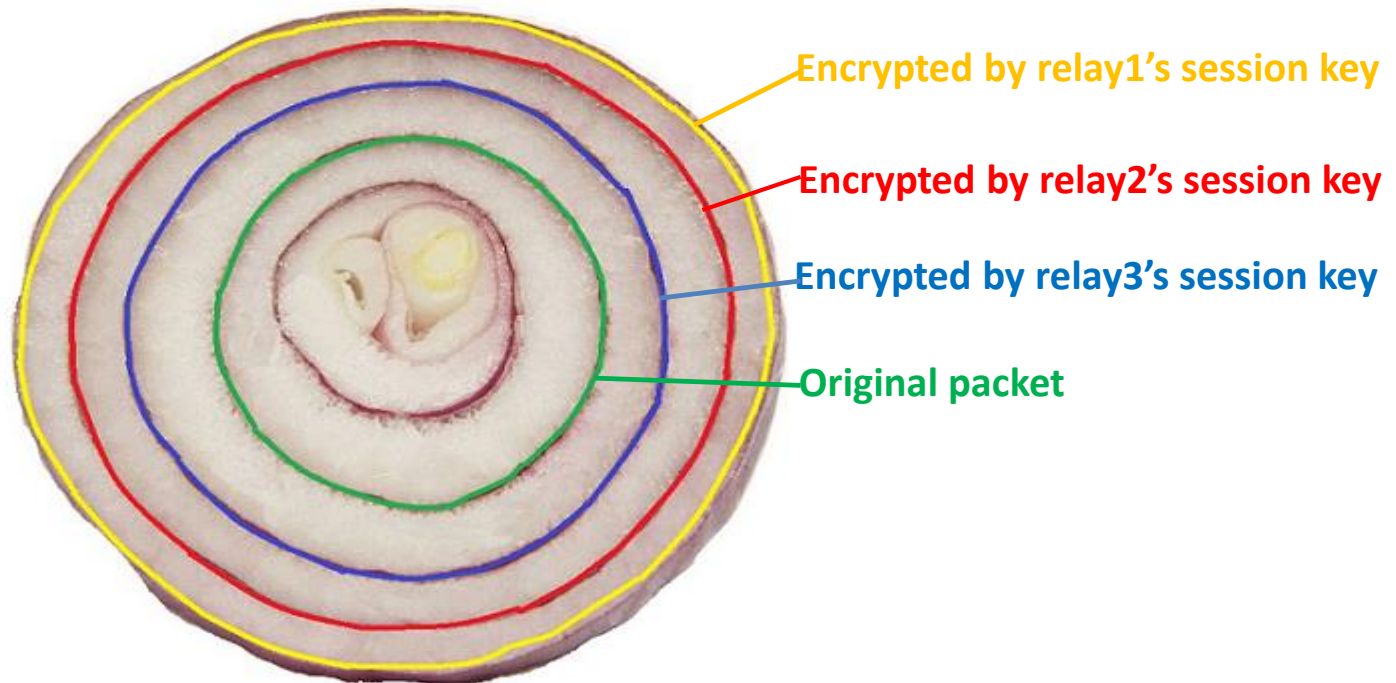
1. Tor client **randomly selects 3 relays** as entry, middle and exit nodes from the node list provided by a directory node
2. Tor client establishes a connection with the **entry node** using its public key and agrees on a **session key**
3. Through the entry node, Tor client establishes a connection with the **middle node** using its public key and agrees on a **session key**
4. Through the entry & middle nodes, Tor client establishes a connection with the **exit node** using its public key and agrees on a **session key**

Onion Routing



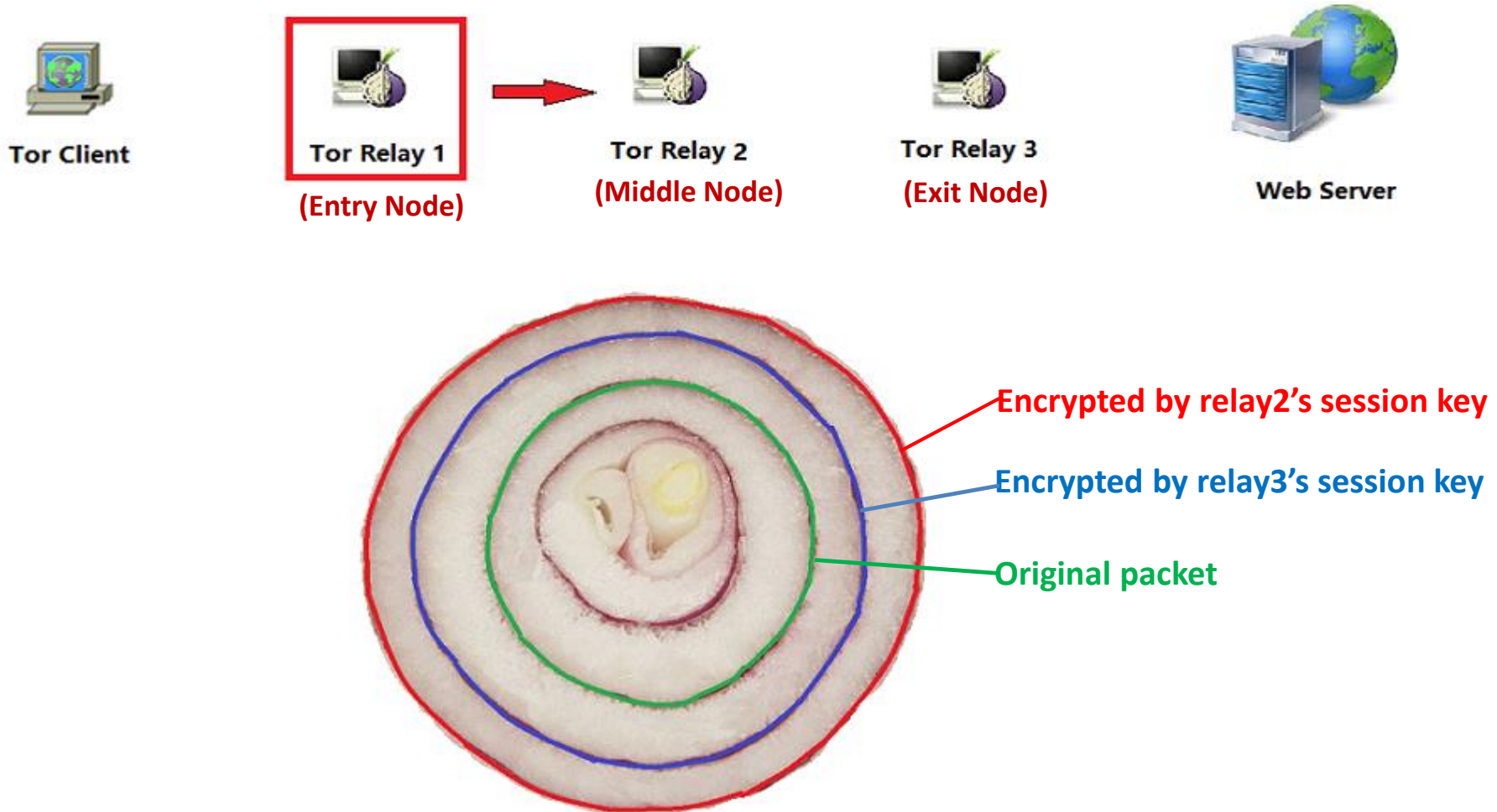
- In above figure, Relay1, Relay2 and Relay3 are chosen to create the circuit.
- Each relay's public IP, port and public key are got from the main Tor directory.
- Packets are encapsulated in layers of encryption just like layers of an onion.

Request Packet Encryption/Decryption (1)



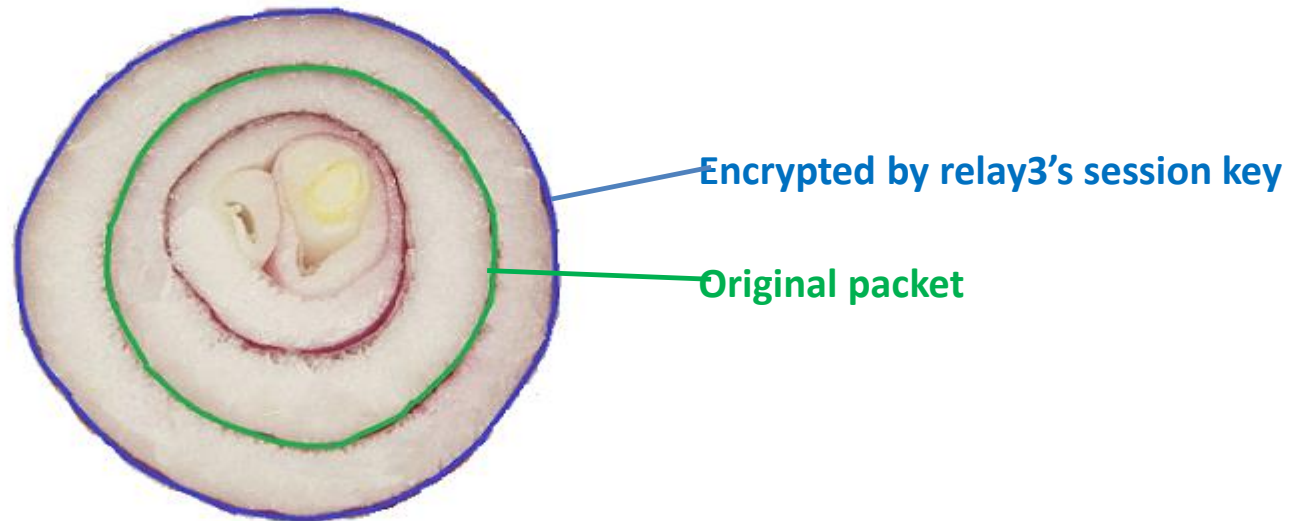
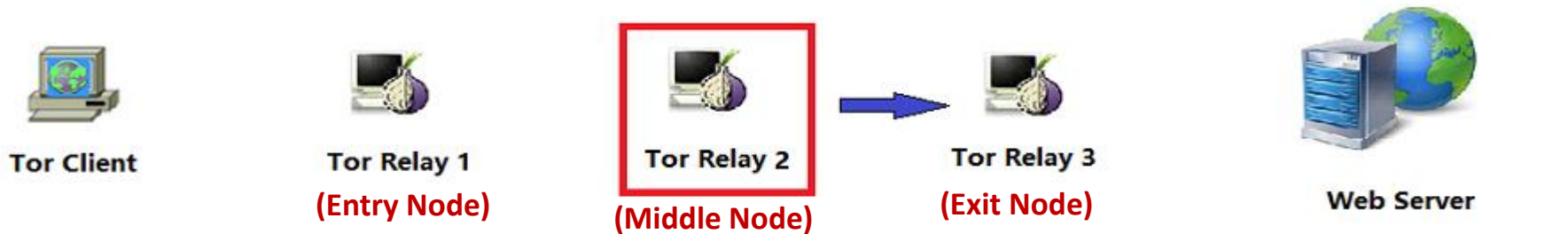
Tor client encrypts the original packet in a three-layered manner with the session key of these 3 relays from the farthest to the nearest, then sends it to the entry node.

Request Packet Encryption/Decryption (2)



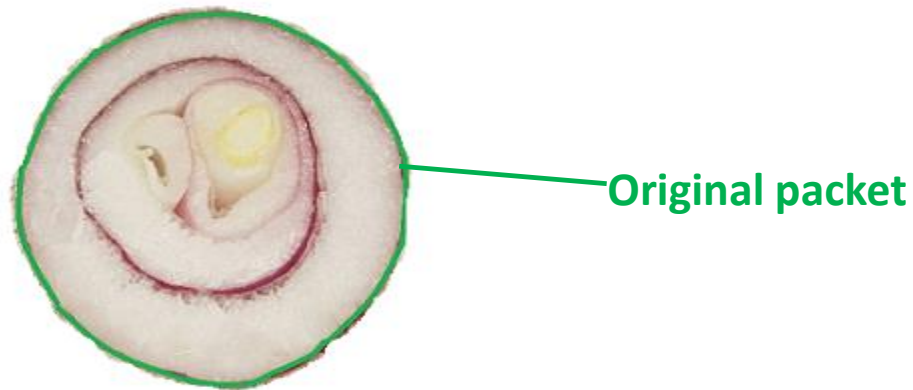
The entry node decrypts the packet with its session key and gets the info of the middle node, then sends the decrypted packet to the middle node.

Request Packet Encryption/Decryption (3)



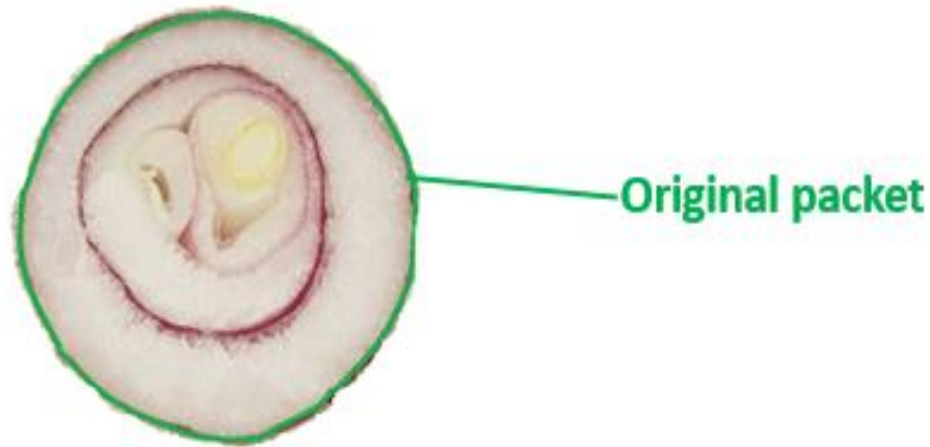
The middle node decrypts the packet with its session key and gets the info of the exit node, then sends the decrypted packet to the exit node.

Request Packet Encryption/Decryption (4)



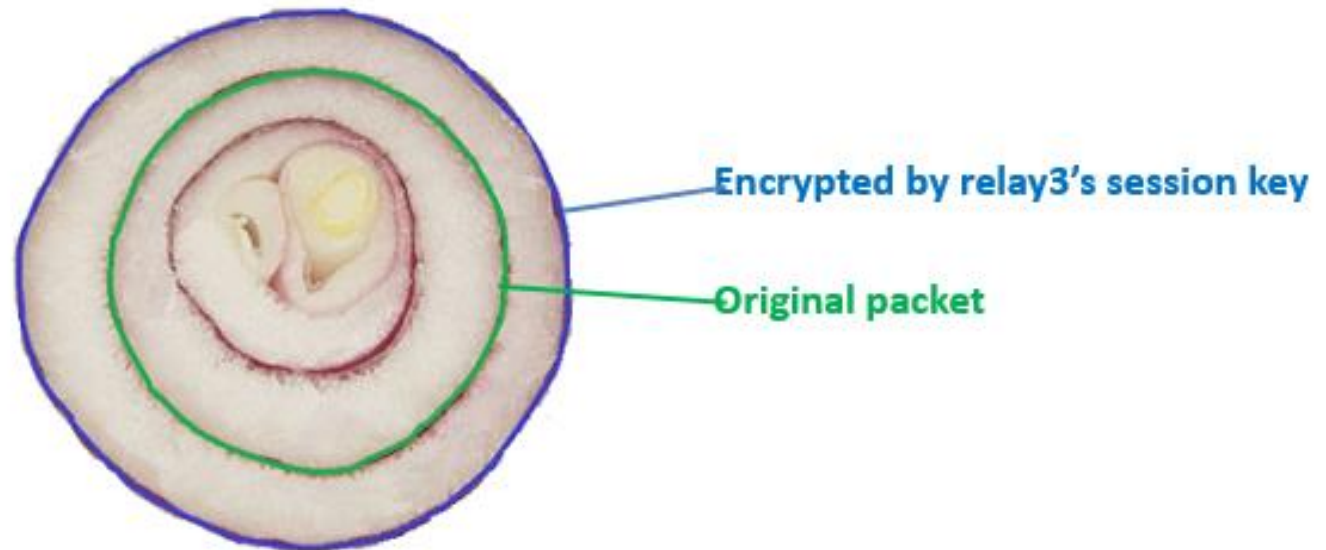
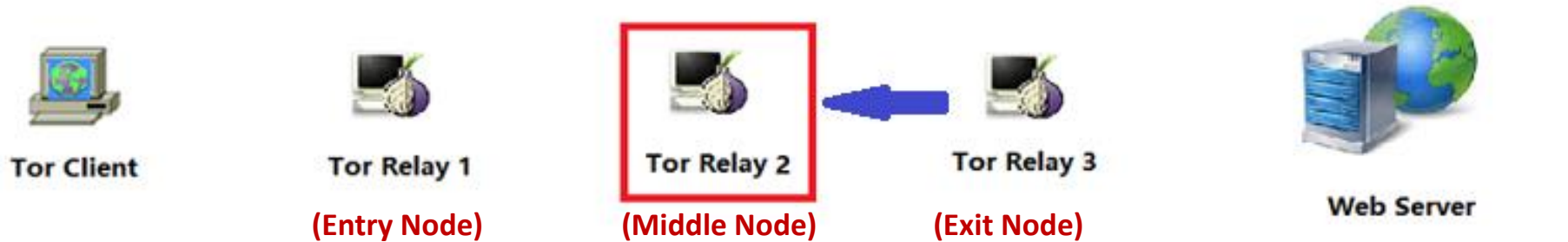
The exit node decrypts the packet with its session key and gets the original packet, then sends it to the destination.

Response Packet Encryption/Decryption



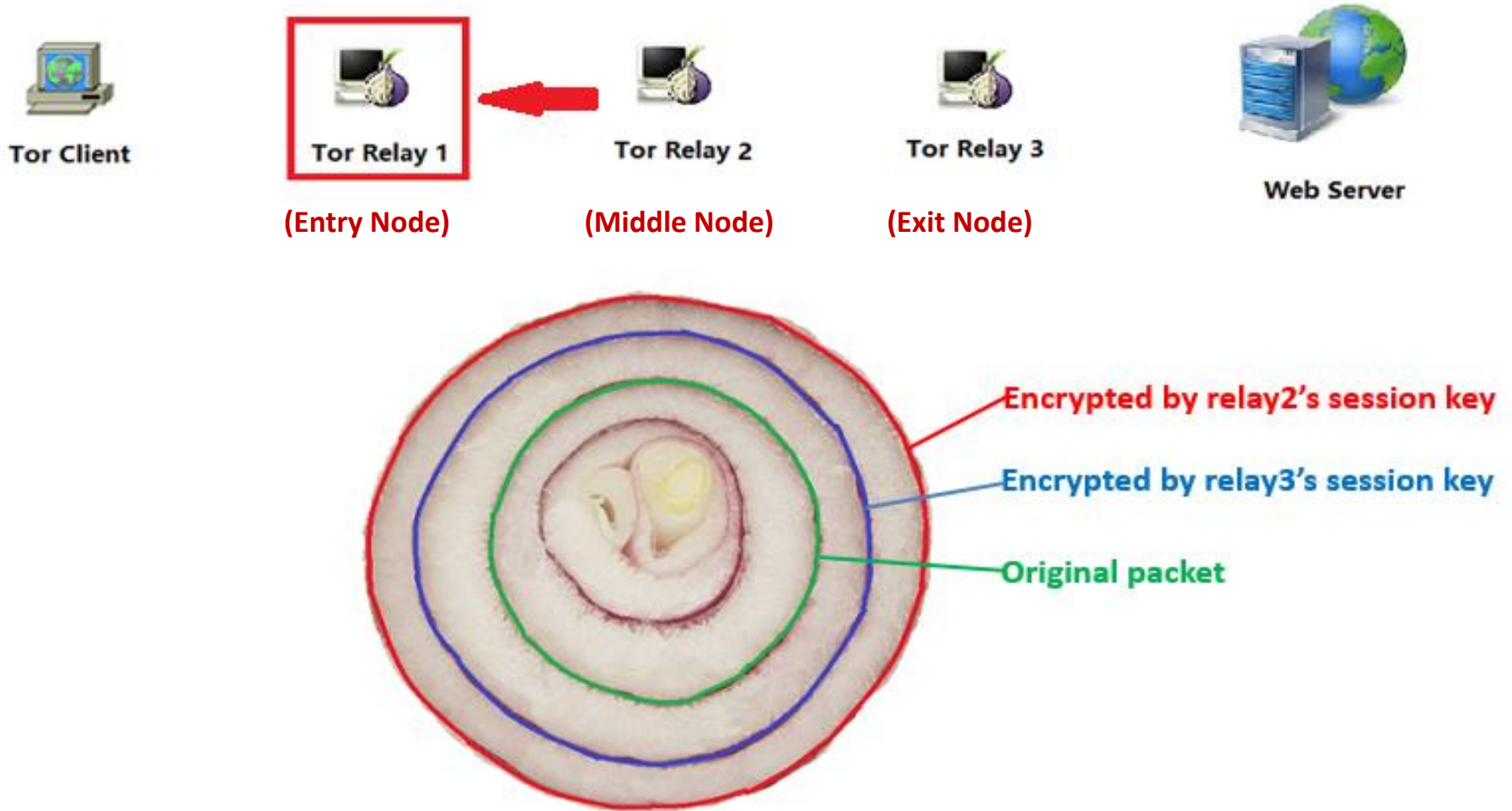
- Each relay encrypts the packet with its session key, then sends it to next relay.
- Tor client receives the packet with 3 layers of encryption, then decrypts it 3 times to get the original packet.

Response Packet Encryption/Decryption



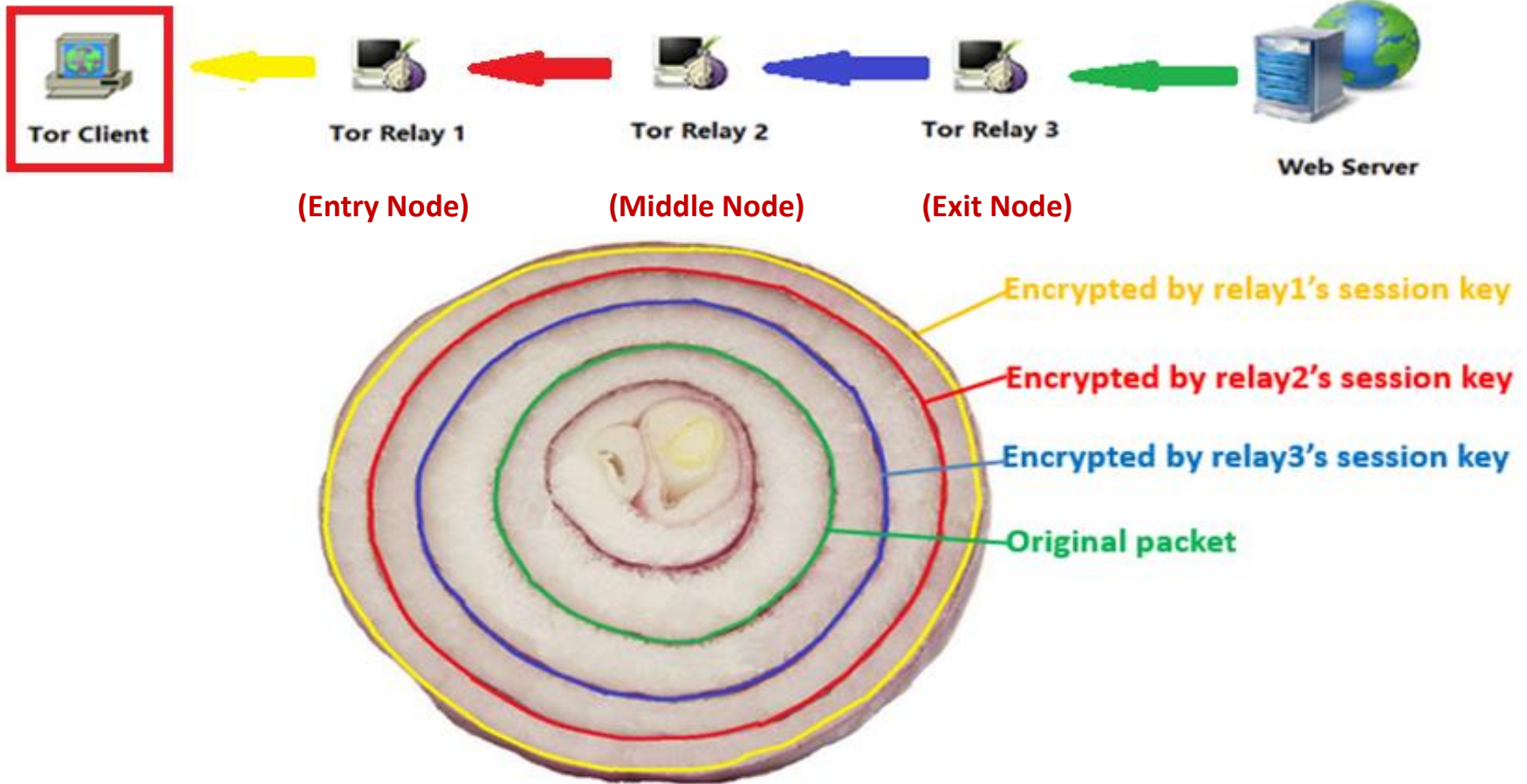
- Each relay encrypts the packet with its session key, then sends it to next relay.
- Tor client receives the packet with 3 layers of encryption, then decrypts it 3 times to get the original packet.

Response Packet Encryption/Decryption



- Each relay encrypts the packet with its session key, then sends it to next relay.
- Tor client receives the packet with 3 layers of encryption, then decrypts it 3 times to get the original packet.

Response Packet Encryption/Decryption



- Each relay encrypts the packet with its session key, then sends it to next relay.
- Tor client receives the packet with 3 layers of encryption, then decrypts it 3 times to get the original packet.

Anonymity

From the above analysis, we can see

- Each relay of a given circuit only knows the previous and next relay
- Only the **Entry** relay knows the Source, but it doesn't know the Destination
- Only the **Exit** relay knows the Destination, but it doesn't know the Source

So Tor network can provide good anonymous communication.

Censorship

But, the normal Tor communication is not resistant to Internet censorship because

- Tor relays are listed in the main Tor directory, so anyone can get them
- Tor traffic uses vanilla Tor protocol which is identifiable

Then, how to solve these issues to circumvent censorship?

Anti-Censorship

Anti-Censorship

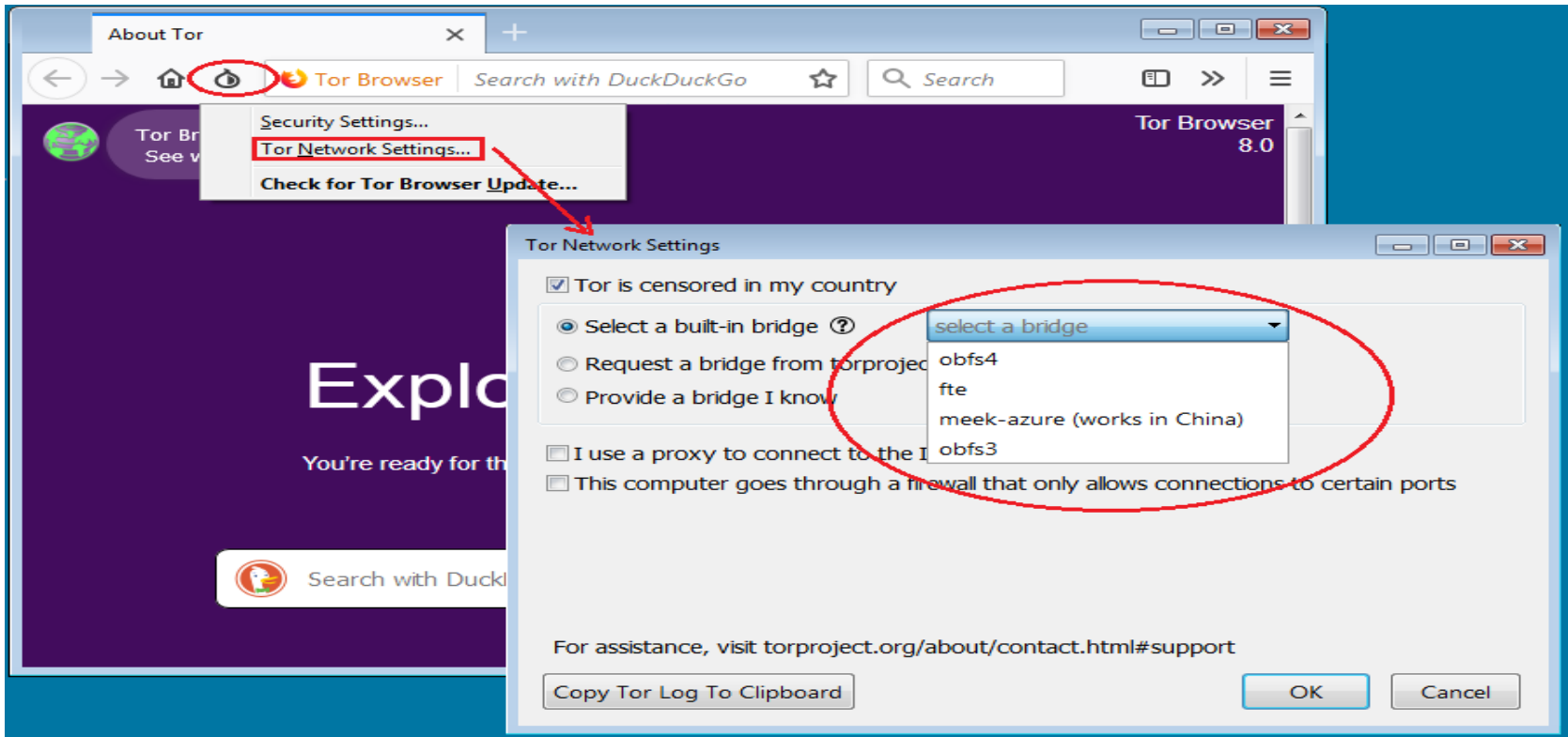
- Tor uses two techniques to circumvent sophisticated censorship
 - ❑ Pluggable Transport
 - ❑ Bridges

Pluggable Transport (PT)

- PT manipulates all Tor traffic between the client and its first relay so that it's not identifiable as Tor traffic.
- Tor supports these PTs: **Obfs3, Obfs4, FTE, Meek and ScrambleSuit**

Pluggable Transport (PT)

- PT manipulates all Tor traffic between the client and its first relay so that it's not identifiable as Tor traffic.
- Tor supports these PTs: **Obfs3, Obfs4, FTE, Meek and ScrambleSuit**



Bridges

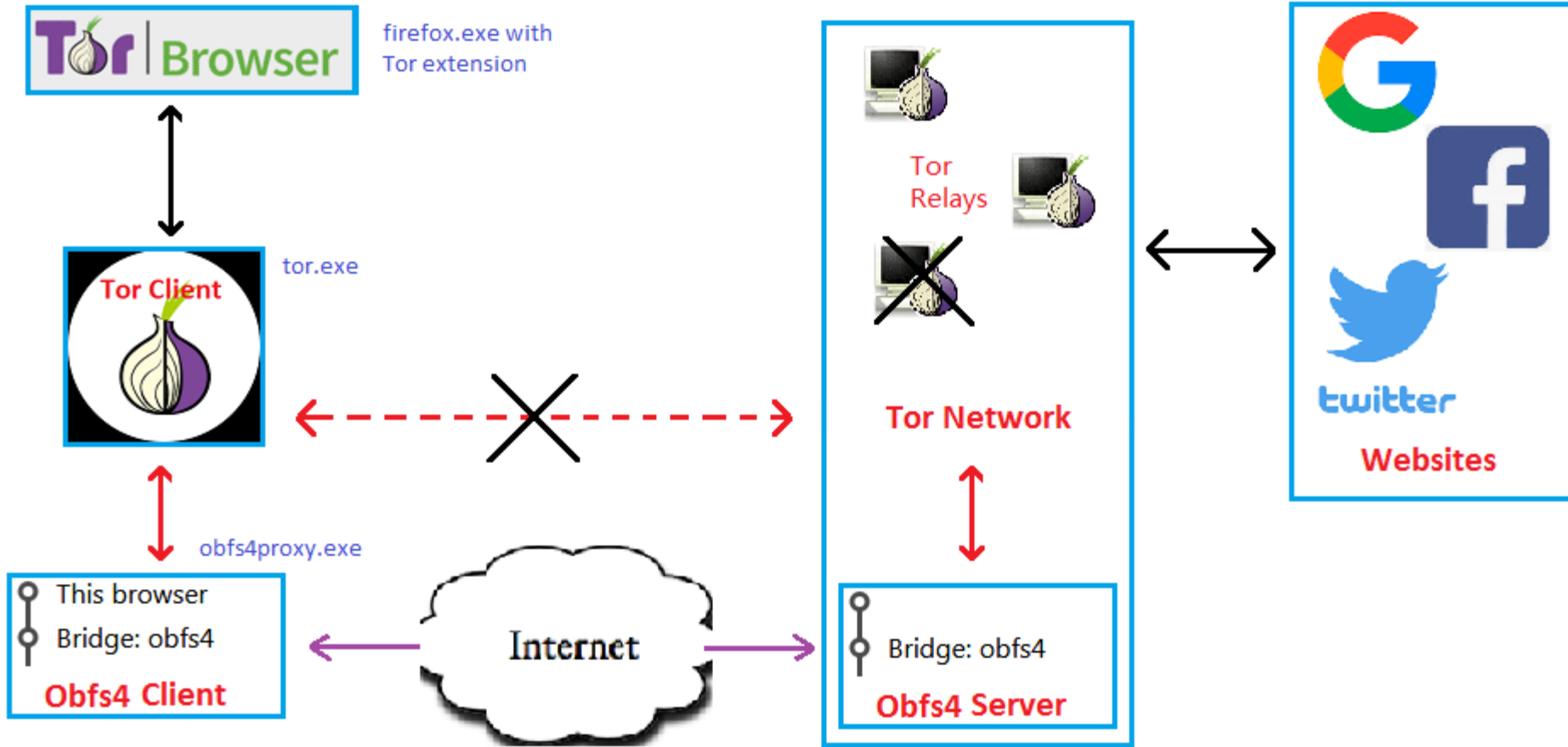
- Bridge relays (or “Bridges” for short) are sort of Tor relays that are **not listed** in the main Tor directory
- There is **no easy way** to get the complete list of the Tor Bridges
- Nobody can block all the Tor Bridges by IP and Port

Obfs4 Bridge



- Obfs4, an **obfuscator**, was developed and maintained by Yawning Angel. It is an open source project written in Go language.
- Obfs4 is not like Obfs3, but is much closer to ScrambleSuit.
- Obfs4 is **strongly recommended** on Tor website.
- Tor browser comes with some default **built-in** Obfs4 bridges.

Tor With Obfs4 Bridge Flow Chart



	normal traffic		traffic is transformed by Obfs4 Bridge
	traffic is encrypted by Tor		tor circuit without Obfs4

The Built-In Obfs4 Bridges

Relationship of The Tor Processes

- ❖ “firefox.exe” (TorLauncher) starts “tor.exe”
- ❖ “tor.exe” starts “obfs4proxy.exe”
- ❖ “obfs4proxy.exe”’s task is to communicate with Obfs4 Bridge relays

Relationship of The Tor Processes

Process Tree

Only show processes still running at end of current trace
 Timelines cover displayed events only

Process	Description	Life Time	Company
[-] Explorer.EXE (1452)	Windows Explorer		Microsoft Corporation
[-] VB_oxTray.exe (2204)	VirtualBox Guest ...		Oracle Corporation
[-] proc_exp.exe (368)	Sysinternals Proce...		Sysinternals - www.sysinternals.c
[-] taskmgr.exe (5584)	Windows Task M...		Microsoft Corporation
[-] Proc_mon.exe (5424)	Process Monitor		Sysinternals - www.sysinternals.c
[-] firefox.exe (836)	Tor Browser		Mozilla Corporation
[-] tor.exe (4220)	n/a		n/a
[-] obfs4proxy.exe (5980)	n/a		n/a

Description: Tor Browser
 Company: Mozilla Corporation
 Path: C:\Users\ \Desktop\Tor Browser\Browser\firefox.exe
 Command: "C:\Users\ \Desktop\Tor Browser\Browser\firefox.exe"
 User:
 PID: 836 Started: 8/8/2019 10:29:54 AM

Go To Event Include Process Include Subtree Close

Find The Built-In Obfs4 Bridges By RE (1)

- Are the built-in Obfs4 bridges hardcoded in “obfs4proxy.exe”?
- Trace from **MSAFD_ConnectEx()** of mswsock.dll

Find The Built-In Obfs4 Bridges By RE (1)

OllyICE - obfs4proxy.exe - [*C.P.U* - thread 000010FC, module obfs4pro]

File View Debug Plugins Options Window Help

Paused

0044ACE6	shl	eax, 2	
0044ACE9	sub	esp, eax	
0044ACEB	mov	edi, esp	
0044ACED	mov	esi, dword ptr [ebx+8]	
0044ACF0	cld		
0044ACF1	rep	movs dword ptr es:[edi], dword ptr [esi]	
0044ACF3	call	dword ptr [ebx]	mswsock.750A7842
0044ACF5	mov	esp, ebp	
0044ACF7	mov	ebx, dword ptr [esp+4]	
0044ACFB	mov	dword ptr [ebx+C], eax	
0044ACFE	mov	dword ptr [ebx+10], edx	
0044AD01	mov	eax, dword ptr fs:[34]	
0044AD08	mov	dword ptr [ebx+14], eax	
0044AD0B	ret		
0044AD0C	int3		
0044AD0D	int3		
0044AD0E	int3		
0044AD0F	int3		

Registers (FPU)

EAX	0000001C
ECX	00000000
EDX	116C98A0
EBX	116C2730
ESP	31D4FF20
EBP	31D4FF3C
ESI	116C98F8
EDI	31D4FF3C
EIP	0044ACF3 obfs4pro.00
C 0	ES 0023 32bit 0(FFF
P 0	CS 001B 32bit 0(FFF

ds:[116C2730]=750A7842 (mswsock.750A7842)

Port 443

1172CB08	02 00	01 BB C0 5F 24 8E	00 00 00 00 00 00 00 00	31D4FF20	00000148
1172CB18	00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	31D4FF24	1172CB08
1172CB28	02 00	Binary IP "192.95.36.142"	00 00 00 00 00 00 00 00	31D4FF28	00000010
1172CB38	00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	31D4FF2C	00000000
1172CB48	00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	31D4FF30	00000000
1172CB58	00 00	00 00 00 00 A0 B0 6A 11	00 00 00 00 00 00 00 00	31D4FF34	00000000
1172CB68	00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	31D4FF38	116B9710
1172CB78	00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	31D4FF3C	004499F4 RETURN to obfs4pro.004499F4

The second arg

Command: ESP EBP NONE

Start:1172CB2D End:1172CB2D Value:0

Find The Built-In Obfs4 Bridges By RE (2)

- The Bridge IP address and Port come from its parent process **tor.exe** over SOCKS5 on the loopback interface (127.0.0.1).
- The IP address and Port of an Obfs4 Bridge are processed in an **event callback function**.

Find The Built-In Obfs4 Bridges By RE (2)

The screenshot shows the OllyICE debugger interface for the process `tor.exe`. The main window displays assembly code for the `WS2_32` module. The registers window shows the state of the CPU registers, with `EAX` containing `764E6C19` and `WS2_32.send`. The disassembly window shows a `CALL` instruction at address `0027F41C` calling `send` from `tor.01521368`. The call stack window shows the current call stack, with the top frame being `Includes WS2_32.send` called from `tor.01521368`. A blue circle highlights the `tor.013734C3` entry in the call stack, which is the address of the `libevent.70F9339D` library. The status bar at the bottom shows the selected range of instructions: `Start:3730890 End:3730899 Sel:0xA`.

Registers (FPU)

EAX	764E6C19	WS2_32.send
ECX	0000000A	
EDX	03730890	
EBX	030CB7A4	
ESP	0027F41C	
EBP	0027F468	
ESI	00063318	
EDI	00000002	

Binary IP and Port

03730890	05 01 00 01 C0 5F 24 8E 01 BB 51 30 73 72	0027F41C	0152136A	CALL to send from tor.01521368
037308A0	4A 49 2F 76 4F 36 56 36 6D 2F 3E 54 61 6E	0027F420	000001A0	Socket = 1A0
037308B0	69 4A 44 33 51 50 32 48 67 7A 55 4B 51 74	0027F424	03730890	Data = 03730890
037308C0				Size = A (10.)
037308D0				s = 0

Call stack of main thread

Address	Procedure / arguments	Called from
0027F41C	Includes WS2_32.send	tor.01521368
0027F420	Socket = 1A0	
0027F424	Data = 03730890	
0027F428	DataSize = A (10.)	
0027F42C	Flags = 0	
0027F46C	tor.01521302	tor.01521687
0027F4BC	tor.01521483	tor.0139F919
0027F53C	? tor.0139F1B0	tor.0139FC72
0027F57C	tor.0139FC37	tor.013734BE
0027F5CC	Includes tor.013734C3	libevent.70F9339D
0027F64C	libevent.70F93131	libevent.70F93662
0027F66C	libevent.70F933B2	libevent.70F93BF2
0027F72C	libevent.70F93AD5	libevent.70F94239
0027F78C	jmp.&libevent-2-1-6.event_base	tor.01376CF6
0027F7DC	tor.01376C09	tor.01376E8E
0027F7FC	tor.01376E77	tor.01376BF2
0027F84C	tor.0137672F	tor.0137B97D
0027F8BC	tor.0137B7A1	tor.01371725
0027F8FC	tor.01371682	tor.01371576

Start:3730890 End:3730899 Sel:0xA

Find The Built-In Obfs4 Bridges By RE (2)

- The Bridge IP address and Port come from its parent process **tor.exe** over SOCKS5 on the loopback interface (127.0.0.1).
- The IP address and Port of an Obfs4 Bridge are processed in an **event callback function**.
- By reverse tracing the IP&Port in “tor.exe”, I finally found a bunch of **Obfs4 Bridge nodes** in a data structure of the command “SETCONF” as its body.

Find The Built-In Obfs4 Bridges By RE (2)

00384184	8B85 60FFFFFF	mov	eax, dword ptr [ebp-A0]	EDX	00000000
0038418A	894424 04	mov	dword ptr [esp+4], eax	EBX	00000000
0038418E	8B85 24FFFFFF	mov	eax, dword ptr [ebp-DC]	ESP	0027F200
00384194	890424	mov	dword ptr [esp], eax	EBP	0027F308
00384197	E8 2532FFFF	call	003773C1	ESI	00823230
0038419C	85C0	test	eax, eax	EDI	00000002

003773C1=003773C1

030CF458	53 45 54 43 4F 4E 46 20 55 73 65 42 72 69 64 67	SETCONF UseBridg	0027F110	00008008	
030CF468	65 73 3D 31 20 42 72 69 64 67 65 3D 22 6F 62 66	es=1 Bridge="obf	0027F114	00000000	
030CF478	73 34 20 31 35 34 2E 33 35 2E 00 00 00 31 31 3A	s4 154.35.22.11:	0027F118	030CF440	
030CF488	38 30 20 41 38 33 32 44 31 37 00 00 00 44 35 48	80 A832D176ECD5C	0027F11C	C3152A49	
030CF498	37 43 36 42 35 38 38 32 35 41 00 00 00 46 43 34	7C6B58825AE22FC4	0027F120	030CF458	ASCII "SETCONF
030CF4A8	43 39 30 46 41 32 34 39 36 33 37 20 63 65 72 74	C90FA249637 cert	0027F124	0000000A	
030CF4B8	3D 59 50 62 51 71 58 50 69 71 54 55 42 66 6A 47	=YPbQqXPiqTUBfjG	0027F128	0027F168	
030CF4C8	46 4C 70 6D 39 4A 59 45 46 54 42 76 6E 7A 45 4A	FLpm9JYEFTBvnzEJ	0027F12C	004F1CD1	tor.004F1CD1
030CF4D8	44 4B 4A 78 58 47 35 53 78 7A 72 72 2F 76 32 71	DKJxXG5Sxzrr/v2q	0027F130	02E83328	
030CF4E8	72 68 47 55 34 4A 6C 73 39 6C 48 6A 4C 41 68 71	rhGU4Jls91HjLAhq	0027F134	00001126	
030CF4F8	70 58 61 45 66 5A 77 20 69 61 74 2D 6D 6F 64 65	pXaEfZw iat-mode	0027F138	00001126	
030CF508	3D 30 22 20 42 72 69 64 67 65 3D 22 6F 62 66 73	=0" Bridge="obfs	0027F13C	02E83328	
030CF518	34 20 31 35 34 2E 33 35 2E 32 32 2E 31 32 3A 34	4 154.35.22.12:4	0027F140	00000000	
030CF528	33 30 34 20 30 30 44 43 36 43 34 46 41 34 39 41	304 00DC6C4FA49A	0027F144	00001126	
030CF538	36 35 42 44 31 34 37 32 39 39 33 43 46 36 37 33	65BD1472993CF673	0027F148	02F2B368	ASCII "SETCONF"
030CF548	30 44 35 34 46 31 31 45 30 44 42 42 20 63 65 72	0D54F11E0DBB cer	0027F14C	02E83328	
030CF558	74 3D 4E 38 36 45 39 68 4B 58 58 58 56 7A 36 47	t=N86E9hXXXVz6G	0027F150	00000000	

Start:30CF560 End:30CF560 Value:58585848

Find The Built-In Obfs4 Bridges By RE (3)

- Loaded automatically from a **local profile file** by Firefox when it starts, and parsed later by TorLauncher.

Find The Built-In Obfs4 Bridges By RE (3)

```
355 pref("extensions.torlauncher.default_bridge.obfs3.2", "obfs3 169.229.59.74:31493 AF9F66B7B04F8FF61 ^
356 pref("extensions.torlauncher.default_bridge.obfs3.3", "obfs3 169.229.59.75:46328 AF9F66B7B04F8FF61
357 pref("extensions.torlauncher.default_bridge.obfs3.4", "obfs3 109.105.109.163:38980 1E05F577A0EC02
358 pref("extensions.torlauncher.default_bridge.obfs3.5", "obfs3 109.105.109.163:47779 4C331FA9B3D1D61
359
360 pref("extensions.torlauncher.default_bridge.fte.1", "fte 131.252.210.150:8080 0E858AC201BF0F3FA3C
361 pref("extensions.torlauncher.default_bridge.fte.2", "fte 128.105.214.161:8080 1E326AAF3FCB515015:
362 pref("extensions.torlauncher.default_bridge.fte.3", "fte 128.105.214.162:8080 FC562097E1951DCC41B
363 pref("extensions.torlauncher.default_bridge.fte.4", "fte 128.105.214.163:8080 A17A40775FBD2CA11841
364
365 pref("extensions.torlauncher.default_bridge.obfs4.1", "obfs4 154.35.22.10:1234 8FB9F4319E89E5C622:
366 pref("extensions.torlauncher.default_bridge.obfs4.2", "obfs4 192.99.11.54:1234 7B126FAB960E5AC6A6:
367 pref("extensions.torlauncher.default_bridge.obfs4.3", "obfs4 109.105.109.165:1234 8DFCD8FB3285E85:
368 pref("extensions.torlauncher.default_bridge.obfs4.4", "obfs4 83.212.101.3:50002 A09D536DD1752D5421
369 pref("extensions.torlauncher.default_bridge.obfs4.5", "obfs4 109.105.109.147:13764 BBB28DF0F201E70
370 pref("extensions.torlauncher.default_bridge.obfs4.6", "obfs4 154.35.22.11:16488 A832D176ECD5C7C6B:
371 pref("extensions.torlauncher.default_bridge.obfs4.7", "obfs4 154.35.22.12:80 00DC6C4FA49A65BD1472:
372 pref("extensions.torlauncher.default_bridge.obfs4.8", "obfs4 154.35.22.13:443 FE7840FE1E21FE0A063:
373 pref("extensions.torlauncher.default_bridge.obfs4.9", "obfs4 154.35.22.10:80 8FB9F4319E89E5C62230:
374 pref("extensions.torlauncher.default_bridge.obfs4.10", "obfs4 154.35.22.10:443 8FB9F4319E89E5C622:
375 pref("extensions.torlauncher.default_bridge.obfs4.11", "obfs4 154.35.22.11:443 A832D176ECD5C7C6B5:
376 pref("extensions.torlauncher.default_bridge.obfs4.12", "obfs4 154.35.22.11:80 A832D176ECD5C7C6B58:
377 pref("extensions.torlauncher.default_bridge.obfs4.13", "obfs4 154.35.22.9:12166 C73ADBAC8ADFDBF0FC0:
378 pref("extensions.torlauncher.default_bridge.obfs4.14", "obfs4 154.35.22.9:80 C73ADBAC8ADFDBF0FC0F:
379 pref("extensions.torlauncher.default_bridge.obfs4.15", "obfs4 154.35.22.9:443 C73ADBAC8ADFDBF0FC01
```

Find The Built-In Obfs4 Bridges By RE (3)

- Loaded automatically from a **local profile file** by Firefox when it starts, and parsed later by TorLauncher.
- “SETCONF” command body was generated with all built-in Obfs4 Bridge information by TorLauncher that runs in firefox.exe.
- It was then sent to tor.exe via a **control port** on loopback interface.

Find The Built-In Obfs4 Bridges By RE (3)

Wireshark · Follow TCP Stream (tcp.stream eq 1)

```

SETCONF UseBridges=1 Bridge="obfs4 154.35.22.9:80 C73ADBAC8ADDFBF0FC0F3F4E8091C0107D093716
cert=gEGKc5WN/bSjFa6UkG9hOcft1tuK+cV8hbZ0H6cqXIMPLqSbCh2Q3PHe50Or6oMVORhoJA iat-mode=0"
Bridge="obfs4 154.35.22.10:443 8FB9F4319E89E5C6223052AA525A192AFBC85D55
cert=GGGS1TX4R81m3r0HBI79wKy1OtPPNR2CZUIrHjkRg65Vc2VR8fOyo64f9kmT1UAFG7j0HQ iat-mode=0"
Bridge="obfs4 38.229.1.78:80 C8CBDB2464FC9804A69531437BCF2BE31FDD2EE4
cert=Hmyfd2ev46gGY7NoVxA9ngrPF2zCZtzskRTzoWXbxNkzeVnGFPWmrTtILRyqCTjHR+s9dg iat-mode=1"
Bridge="obfs4 38.229.33.83:80 0BAC39417268B96B9F514E7F63FA6FBA1A788955 cert=VwEFpk9F/
UN9JED7XpG1XOjm/O8ZCXK80oPecgWnNDZDv5pdkhq1OpbAH0wNqOT6H6BmRQ iat-mode=1" Bridge="obfs4
192.95.36.142:443 CDF2E852BF539B82BD10E27E9115A31734E378C2 cert=qUVQ0srL1JI/vO6V6m/
24anYXjD3QP2HgzUKQtQ7GRqqUvs7P+tG43RtAqdhLOALP7DJQ iat-mode=1" Bridge="obfs4 154.35.22.10:15937
8FB9F4319E89E5C6223052AA525A192AFBC85D55
cert=GGGS1TX4R81m3r0HBI79wKy1OtPPNR2CZUIrHjkRg65Vc2VR8fOyo64f9kmT1UAFG7j0HQ iat-mode=0"
Bridge="obfs4 109.105.109.147:13764 BBB28DF0F201E706BE564EFE690FE9577DD8386D cert=KfMQN/
tNMFdda61hMgpiMI7pbwU1T+wxjTulYnfw+4sgvG0zSH7N7fwT10BI8MUdAD7jA iat-mode=2" Bridge="obfs4
154.35.22.12:80 00DC6C4FA49A65BD1472993CF6730D54F11E0DBB cert=N86E9hKXXXVz6G7w2z8wFfhIDztDazZ/
3poxVePHEYjbKDWzjkRDccFMANhK75fc65pYSg iat-mode=0" Bridge="obfs4 85.31.186.26:443
91A6354697E6B02A386312F68D82CF86824D3606 cert=PBwr+S8JTVZo6MPdHnkTwXJPILWADLqfMGoVvhZCIMq/
Urndyd42BwX9YFJHZnBB3H0XCw iat-mode=0" Bridge="obfs4 37.218.245.14:38224
D9A82D2F9C2F65A18407B1D2B764F130847F8B5D
cert=bjRaMrr1BRiAW8IE9U5z27fQaYgOhX1UCmOpg2pFpoMvo6ZgQMzLsaTzzQNTlm7hNcb+Sg iat-mode=0"
    
```

4,225 client pkts, 0 server pkts, 0 turns.

Entire conversation (4225 bytes) Show and save data as ASCII Stream 1

Find: Find Next

Filter Out This Stream Print Save as... Back Close Help

Find The Built-In Obfs4 Bridges By RE (3)

about:config

Tor ...ser about:config

Search:

Preference Name	Status	Type	Value
extensions.torlauncher.default_bridge.obfs4.1	default	string	obfs4 154.35.22.10:15937 8FB9F4319E89E5C6223052...
extensions.torlauncher.default_bridge.obfs4.10	default	string	obfs4 154.35.22.10:443 8FB9F4319E89E5C6223052AA...
extensions.torlauncher.default_bridge.obfs4.11	default	string	obfs4 154.35.22.11:443 A832D176ECD5C7C6B58825...
extensions.torlauncher.default_bridge.obfs4.12	default	string	obfs4 154.35.22.11:80 A832D176ECD5C7C6B58825A...
extensions.torlauncher.default_bridge.obfs4.13	default	string	obfs4 154.35.22.9:12166 C73ADBAC8ADFD8BF0FC0F3...
extensions.torlauncher.default_bridge.obfs4.14	default	string	obfs4 154.35.22.9:80 C73ADBAC8ADFD8BF0FC0F3F4E...
extensions.torlauncher.default_bridge.obfs4.15	default	string	obfs4 154.35.22.9:443 C73ADBAC8ADFD8BF0FC0F3F4...
extensions.torlauncher.default_bridge.obfs4.16	default	string	obfs4 154.35.22.12:4304 00DC6C4FA49A65BD147299...
extensions.torlauncher.default_bridge.obfs4.17	default	string	obfs4 154.35.22.13:16815 FE7840FE1E21FEC0A0639ED...
extensions.torlauncher.default_bridge.obfs4.18	default	string	obfs4 192.95.36.142:443 CDF2E852BF539B32BD10E2...
extensions.torlauncher.default_bridge.obfs4.19	default	string	obfs4 85.17.30.79:443 FC259A04A328A07FED1413E9...
extensions.torlauncher.default_bridge.obfs4.2	default	string	obfs4 192.99.11.54:443 7B126FAB960E5AC6A629C72...
extensions.torlauncher.default_bridge.obfs4.20	default	string	obfs4 38.229.1.78:80 C8CBDB2484FC9804A69531437...
extensions.torlauncher.default_bridge.obfs4.21	default	string	obfs4 38.229.33.83:80 0BAC39417268B96B9F514E7F6...
extensions.torlauncher.default_bridge.obfs4.22	default	string	obfs4 [2001:476:b381:bfff:216:3eff:fe23:d6c3]:443 C...
extensions.torlauncher.default_bridge.obfs4.23	default	string	obfs4 37.218.240.34:40035 88CD36D45A35271963EF8...
extensions.torlauncher.default_bridge.obfs4.24	default	string	obfs4 37.218.245.14:38224 D9A82D2F9C2F65A18407...
extensions.torlauncher.default_bridge.obfs4.25	default	string	obfs4 85.31.186.98:443 011F2599C0E9B27EE74B3531...

How Tor Client Connects To Obfs4 Bridge

Tor Browser (Firefox) Starts With Obfs4

- Extensions TorLauncher and TorButton
- TorLauncher starts **tor.exe (Tor Client)**

Tor Browser (Firefox) Starts With Obfs4

```
544
545     if (!path && !torFile)
546     {
547         // No preference and no pre-determined IPC path: use a default path.
548         isRelativePath = true;
549         if (TLUtilInternal._isUserDataOutsideOfAppDir)
550         {
551             // This block is used for the TorBrowser-Data/ case.
552             if (this.isWindows)
553             {
554                 if ("tor" == aTorFileType)
555                     path = "TorBrowser\\Tor\\tor.exe";
556                 else if ("pt-startup-dir" == aTorFileType)
557                     useAppDir = true;
558                 else if ("torrc-defaults" == aTorFileType)
559                     path = "TorBrowser\\Tor\\torrc-defaults";
560                 else if ("torrc" == aTorFileType)
561                     path = "Tor\\torrc";
562                 else if ("tordatadir" == aTorFileType)
563                     path = "Tor";
564             }
565             else if (this.isMac)
566             {
567                 if ("tor" == aTorFileType)
568                     path = "Contents/Resources/TorBrowser/Tor/tor";
569                 else if ("pt-startup-dir" == aTorFileType)
570                     path = "Contents/MacOS/Tor";
571                 else if ("torrc-defaults" == aTorFileType)
572                     path = "Contents/Resources/TorBrowser/Tor/torrc-defaults";
573                 else if ("torrc" == aTorFileType)
```

Tor Listens On Loopback Interface

- Loopback address: **127.0.0.1**
- Tor control port: TCP Port **9151**
- Tor proxy port: TCP Port **9150**

Tor Listens On Loopback Interface

OlllyICE - [*C.P.U* - main thread, module xul]

File View Debug Plugins Options Window Help

Paused

60878F0E	movzx	eax, al		
60878F11	mov	dword ptr [ebp-D4], eax		
60878F17	mov	dword ptr [ebp-EC], 8140		
60878F21	test	esi, esi		
60878F23	je	short 60878F2B		
60878F25	mov	dword ptr [ebp-DC], esi		
60878F2B	lea	eax, dword ptr [ebp-F0]		
60878F31	mov	dword ptr [esp], eax		
60878F34	call	dword ptr [.&SHELL32.ShellExecuteExW]	SHELL32.ShellExecuteExW	
60878F3A	sub	esp, 4		
60878F3D	mov	ebx, 80520003		
60878F42	test	eax, eax		
60878F44	je	6087921C		

Registers (FPU)

EAX	0038E0D8
ECX	00000000
EDX	0038E0D8
EBX	065947D8
ESP	0038E070
EBP	0038E1C8
ESI	09FE1240 UNICODE
EDI	0038E114
EIP	60878F34 xul.60878

ds:[65504640]=769B1E65 (SHELL32.ShellExecuteExW)

Command line parameters

09FE1240	--defaults-torrc "C:\Users\MOYTes0Env\Desktop\Tor Browser\Browse	0038E078	09FEOCF8	
09FE12C0	r\TorBrowser\Data\Tor\torrc-defaults" -f "C:\Users\MOYTes0Env\De	0038E07C	FFFFFFFF	
09FE1340	esktop\Tor Browser\Browser\TorBrowser\Data\Tor\torrc" DataDirecto	0038E080	09FE1240	UNICODE "--defa
09FE13C0	ry "C:\Users\MOYTes0Env\Desktop\Tor Browser\Browser\TorBrowser\D	0038E084	000002A0	
09FE1440	ata\Tor" GeoIPFile "C:\Users\MOYTes0Env\Desktop\Tor Browser\Brow	0038E088	777EE325	ntdll.777EE325
09FE14C0	ser\TorBrowser\Data\Tor\geoiP GeoIPV6File "C:\Users\MOYTes0Env\	0038E08C	77822B8F	RETURN to ntdll
09FE1540	Desktop\Tor Browser\Browser\TorBrowser\Data\Tor\geoiP" HashedCo	0038E090	00A48A80	
09FE15C0	ntrolPassword 16:d8944b4380655bb76012c9ebf8eea49c4f8f722e7793036	0038E094	065947D8	
09FE1640	e2130310374 + ControlPort 9151 + _SocksPort "127.0.0.1:9150 IP	0038E098	00000002	
09FE16C0	v6Traffic PreferIPv6 KeepAliveIsolateSOCKSAuth" _OwningControll	0038E09C	000F0000	
09FE1740	erProcess 5012 DisableNetwork 1. 解帯-Å0 .é...→.é...←.é...	0038E0A0	00000000	
09FE17C0	#.N...\$.N..8).δ. 8*.r.↑3/.5. 0.7.↑.5.0.β.6.0.>.9.∩. .=.	0038E0A4	0000FDE9	
09FE1840	..C.t. .D.η.↓.I.Σ. .J.U.↓.0.0.β.P.Y.>.S.I. .W.0. .X.0...].	0038E0A8	09FEOCF8	

M1 M2 M3 M4 M5 Command: ESP EBP NONE

Start:9FE189A End:9FE189B Value:1D50000

Tor Listens On Loopback Interface

01b1ce - [C:\BIP* - main thread - module vull]

Tor Command Line Parameters and Values
--defaults-torrc "...\Browser\TorBrowser\Data\Tor\torrc-defaults"
-f "...\Browser\TorBrowser\Data\Tor\torrc"
DataDirectory "...\Browser\TorBrowser\Data\Tor"
GeoIPFile "...\Browser\TorBrowser\Data\Tor\geoiP"
GeoIPv6File "...\Browser\TorBrowser\Data\Tor\geoiP6"
HashedControlPassword 16:d8944b4380655bb76012c9ebf8eea49c4f8f722e7793036e2130310374
+ __ControlPort 9151
+ __SocksPort "127.0.0.1:9150 IPv6Traffic PreferIPv6 KeepAliveIsolateSOCKSAuth"
__OwningControllerProcess 1748
DisableNetwork 1

"...\\" is short for the Tor Browser's installation path.

Tor Browser Sends SETCONF To Tor

TCPView - Sysinternals: www.sysinternals.com

File Options Process View Help

Proc...	Protocol	Local Address	Remote Address	State
[System Process]0	TCP	127.0.0.1:49553	127.0.0.1:49496	TIME_WAIT
firefox.exe:3092	TCP	127.0.0.1:49488	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49491	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49540	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49548	127.0.0.1:9150	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49512	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49506	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49559	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49511	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49509	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49510	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49519	38.229.33.83:80	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49520	37.218.245.14:38224	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49527	109.105.109.147:13764	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49528	37.218.240.34:40035	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49529	192.95.36.142:443	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49507	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49513	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49502	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49501	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49560	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49561	83.212.101.3:50002	SYN_SENT
obfs4proxv.exe:2624	TCP	10.0.2.15:49562	154.35.22.9:80	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49488	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49491	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49501	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49502	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49506	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49507	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49509	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49510	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49540	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9150	127.0.0.1:49548	ESTABLISHED

Endpoints: 71 Established: 42 Listening: 0 Time Wait: 29 Close Wait: 0

Tor Starts Obfs4Proxy

- Tor parses SETCONF command and starts **obfs4proxy.exe** (**Obfs4Proxy** or **Obfs4 Client**)
- Obfs4Proxy informs Tor of its TCP Port number that listens on loopback through a **inter-process pipe**

Tor Starts Obfs4Proxy

OllyICE - obfs4proxy.exe

File View Debug Plugins Options Window Help

Paused

C.P.U - main thread, module KERNELBA

Address	Disassembly	Comment
759B754E	6A 18	push 18
759B7550	68 D0759B75	push 759B75D0
759B7555	E8 46A1FFFF	call 759B16A0
759B755A	33C9	xor ecx, ecx
759B755C	894D E0	mov dword ptr [ebp-20], ecx
759B755F	33C0	xor eax, eax
759B7561	8D7D E4	lea edi, dword ptr [ebp-1C]
759B7564	AB	stos dword ptr es:[edi]
759B7565	8B7D 14	mov edi, dword ptr [ebp+14]
759B7568	3BF9	cmp edi, ecx
759B756A	74 02	je short 759B756E
759B756C	890F	mov dword ptr [edi], ecx

Registers (FPU)

EAX 00000100
 ECX 116F3CF8
 EDX 00000000
 EBX 007777B0 obfs4pro.007777B0
 ESP 0006FE88
 EBP 0006FEA0
 ESI 116F3CAC
 EDI 0006FEBC
 EIP 759B754E KERNELBA.WriteFile
 C 1 ES 0023 32bit 0(FFFFFFFF)

Address	Disassembly	Comment
116A8470	00 00 00 00
116A8480	43 4D 45 54 48 4F 44 20 6F 62 66 73 32 20 73 6F	CMETHOD obfs2 so
116A8490	63 6B 73 35 20 31 32 37 2E 30 2E 30 2E 31 3A 34	cks5 127.0.0.1:4
116A84A0	39 34 33 37 00 00 00 00 00 00 00 00 00 00 00 00	9437.....
116A84B0	43 4D 45 54 48 4F 44 20 6F 62 66 73 34 20 73 6F	CMETHOD obfs4 so
116A84C0	63 6B 73 35 20 31 32 37 2E 30 2E 30 2E 31 3A 34	cks5 127.0.0.1:4
116A84D0	39 34 39 36 0A 00 00 00 00 00 00 00 00 00 00 00	9496.....
116A84E0	43 4D 45 54 48 4F 44 20 6F 62 66 73 33 20 73 6F	CMETHOD obfs3 so
116A84F0	63 6B	
116A8500	31 39	
116A8510	43 4D	
116A8520	63 6B	
116A8530	31 39	
116A8540	00 00	
116A8550	00 00	
116A8560	00 00	
116A8570	00 00	
116A8580	00 00	
116A8590	00 00	
116A85A0	00 00	
116A85B0	00 00	
116A85C0	00 00	

CALL to WriteFile from kernel32.777456DF
 hFile = 00000100 (window)
 Buffer = 116A84B0
 nBytesToWrite = 25 (37.)
 pBytesWritten = 116F3CF8
 pOverlapped = NULL

RETURN to obfs4pro.0044ACF5

RETURN to obfs4pro.004499F4
 obfs4pro.007777B0

RETURN to obfs4pro.005D7E3D from obfs4pro.004

Handles

Handle	Type	Refs	Access	T	Info	Name
000000E0	File	3.	0012019F			
000000E4	File (pipe)	4.	0016019F			\Device\Afd
000000E8	Event	3.	001F0003			
000000EC	File (pipe)	2.	0016019F			\Device\Afd
000000F4	File (pipe)	4.	0016019F			\Device\Afd
00000100	File (pipe)	3.	00120196			\Device\NamedPipe
00000104	File (pipe)	4.	0016019F			\Device\Afd
00000108	File (pipe)	3.	00120196			\Device\NamedPipe
00000110	File (pipe)	3.	00120189			\Device\NamedPipe

Tor Starts Obfs4Proxy

- Tor parses SETCONF command and starts **obfs4proxy.exe** (**Obfs4Proxy** or **Obfs4 Client**)
- Obfs4Proxy informs Tor of its TCP Port number that listens on loopback through a **inter-process pipe**
- Tor then separately sends the Bridges to that TCP Port

Tor Starts Obfs4Proxy

TCPView - Sysinternals: www.sysinternals.com

File Options Process View Help

Proc...	Protocol	Local Address	Remote Address	State
[System Process]:0	TCP	127.0.0.1:49553	127.0.0.1:49496	TIME_WAIT
firefox.exe:3092	TCP	127.0.0.1:49488	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49491	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49540	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49548	127.0.0.1:9150	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49512	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49506	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49559	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49511	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49509	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49510	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49519	38.229.33.83:80	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49520	37.218.245.14:38224	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49527	109.105.109.147:13764	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49528	37.218.240.34:40035	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49529	192.95.36.142:443	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49507	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49513	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49502	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49501	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49560	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49561	83.212.101.3:50002	SYN_SENT
obfs4proxy.exe:2624	TCP	10.0.2.15:49562	154.35.22.9:80	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49488	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49491	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49501	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49502	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49506	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49507	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49509	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49510	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49540	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9150	127.0.0.1:49548	ESTABLISHED

Endpoints: 71 Established: 42 Listening: 0 Time Wait: 29 Close Wait: 0

Tor separately sends Obfs4 Bridge information to Obfs4Proxy over SOCK5 Protocol.

Obfs4Proxy Connects To Obfs4 Bridges

- Obfs4Proxy uses Bridge information received from Tor to establish connection with Obfs4 Bridge
- Obfs4Proxy sends “05 00 00 01 00 00 00 00 00 00 00” to Tor once connection is established

Obfs4Proxy Connects To Obfs4 Bridges

00000000 05 02 00 02
 00000000 05 02
 00000004 01 56 63 65 72 74 3d 42 76 67 2f 69 74 78 65 4c
 00000014 34 54 57 4b 4c 50 36 4e 31 4d 61 51 7a 53 4f 43
 00000024 36 74 63 52 49 42 76 36 71 35 37 44 59 41 5a 63
 00000034 33 62 32 41 7a 75 4d 2b 2f 54 66 42 37 6d 71 54
 00000044 46 45 66 58 49 4c 43 6a 45 77 7a 56 41 3b 69 61
 00000054 74 2d 6d 6f 64 65 3d 31 01 00
 00000002 01 00
 0000005E 05 01 00 01 6d 69 6d a5 29 1f
 00000004 05 00 00 01 00 00 00 00 00 00
 00000068 16 03 01 00 b8 01 00 00 b4 03 03 5a f8 62 9d 1b
 00000078 67 32 cf aa be 12 1e a3 2c 8f 58 e3 d4 b7 fb 35
 00000088 6e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000098 c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 000000A8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 000000B8 00 16 00 14 00 00 11 77 77 77 2e 6b 6f 63 79 63
 000000C8 68 70 6b 34 2e 63 6f 6d 00 0b 00 04 03 00 01 02
 000000D8 00 0a 00 1c 00 1a 00 17 00 19 00 1c 00 1b 00 18
 000000E8 00 1a 00 16 00 0e 00 0d 00 0b 00 0c 00 09 00 0a
 000000F8 00 23 00 00 00 0d 00 20 00 1e 06 01 06 02 06 03
 00000108 05 01 05 02 05 03 04 01 04 02 04 03 03 01 03 02
 00000118 03 03 02 01 02 02 02 03 00 0f 00 01 01
 0000000E 16 03 03 00 39 02 00 00 35 03 03 a7 40 05 1a c3
 0000001E cb b5 3f a1 90 33 f5 b3 4f 52 ad 37 6c b4 84 72
 0000002E 82 bb 40 bd 34 ce 68 bc 9b 5c dc 00 c0 30 00 00
 0000003E 0d ff 01 00 01 00 00 0b 00 04 03 00 01 02 16 03
 0000004E 03 02 48 0b 00 02 44 00 02 41 00 02 3e 30 82 02
 0000005E 3a 30 82 01 a3 a0 03 02 01 02 02 08 56 09 64 dc

SOCKS5 Protocol
 One Obfs4 Bridge Data
 Connection Established
 Tor Payload
 Obfs4 Bridge IP and Port in binary
 109.105.109.165:10527

9 client pkts, 11 server pkts, 15 turns.

Obfs4Proxy Connects To Obfs4 Bridges

TCPView - Sysinternals: www.sysinternals.com

File Options Process View Help

Proc...	Protocol	Local Address	Remote Address	State
[System Process]:0	TCP	127.0.0.1:49553	127.0.0.1:49496	TIME_WAIT
firefox.exe:3092	TCP	127.0.0.1:49488	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49491	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49540	127.0.0.1:9151	ESTABLISHED
firefox.exe:3092	TCP	127.0.0.1:49548	127.0.0.1:9150	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49512	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49506	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49559	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49511	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49509	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49510	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49519	38.229.33.83:80	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49520	37.218.245.14:38224	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49527	109.105.109.147:13764	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49528	37.218.240.34:40035	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49529	192.95.36.142:443	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49507	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49513	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49502	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49501	ESTABLISHED
obfs4proxy.exe:2624	TCP	127.0.0.1:49496	127.0.0.1:49560	ESTABLISHED
obfs4proxy.exe:2624	TCP	10.0.2.15:49561	83.212.101.3:50002	SYN_SENT
obfs4proxy.exe:2624	TCP	10.0.2.15:49562	154.35.22.9:80	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49488	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49491	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49501	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49502	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49506	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49507	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49509	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:49510	127.0.0.1:49496	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9151	127.0.0.1:49540	ESTABLISHED
tor.exe:2096	TCP	127.0.0.1:9150	127.0.0.1:49548	ESTABLISHED

Endpoints: 71 | Established: 42 | Listening: 0 | Time Wait: 29 | Close Wait: 0

3

Obfs4proxy makes connections to Obfs4 Bridges

Obfs4Proxy Connects To Obfs4 Bridges

- Obfs4Proxy uses Bridge information received from Tor to establish connection with Obfs4 Bridge
- Obfs4Proxy sends “05 00 00 01 00 00 00 00 00 00 00” to Tor once connection is established
- Tor encrypts the proxy data from Firefox, then sends **Tor-encrypted** data to Obfs4Proxy which transforms and transports it to Obfs4 Bridge

How Obfs4 Transforms Tor-Encrypted Traffic

Obfs4 Bridge Configuration Line

```

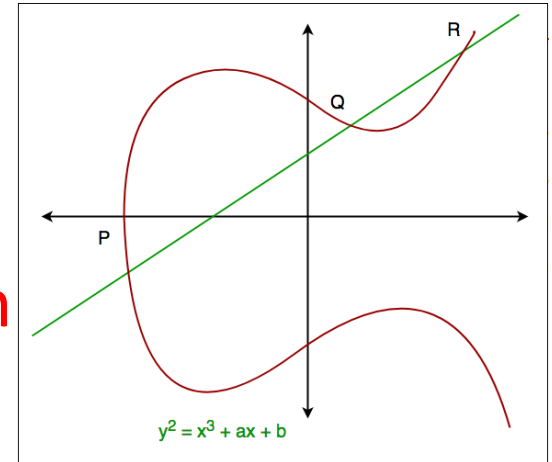
SETCONF UseBridges=1 Bridge="obfs4 109.105.109.165:10527
8DFCD8FB3285E855F5A55EDDA35696C743ABFC4E
cert=Bvg/itxeL4TWKLP6N1MaQzSOC6tcRIBv6q57DYAZc3b2Azum+/TfB7m
qTFEfXILCjEwzVA iat-mode=1" Bridge="obfs4 85.17.30.79:443 .....

```

- “SETCONF” is the command name and followed are all built-in Obfs4 Bridges
- One bridge configuration line contains:
 - Bridge type: obfs4
 - Bridge server IP address and port: 109.105.109.165:10527
 - Bridge ID: 14H long hexadecimal
 - Bridge cert: Base64-encoded **nodeID**, **idPublicKey**, which participate in generating common keySeed
 - Bridge iat-mode: iat mode flag can be “0”, “1” and “2”

Elliptic Curve Cryptography (ECC)

- Obfs4 Bridge uses the **ECC algorithm** to make secure communication
- ECC is a public key encryption technique based on **elliptic curve theory**
- The ECC algorithm Obfs4 used is implemented in **curve25519** package in Go language
- Two functions: `ScalarBaseMult()` and **ScalarMult()**



Obfs4 KeyPair

```
// Keypair is a Curve25519 keypair with an optional Elligator representative.  
// As only certain Curve25519 keys can be obfuscated with Elligator, the  
// representative must be generated along with the keypair.  
type Keypair struct {  
    public      *PublicKey  
    private    *PrivateKey  
    representative *Representative  
}
```

- Both client and server sides must have their own **KeyPair**
- Public Key is computed from Private Key
- Representative can be used to restore Public Key

Obfs4 Client Handshake

Size	Content
20H	Client's representative (Keypair.representative)
variable	Padding data, data size range: 4Dh~1FC0h
10H	mark, HMAC of Client's representative
10H	HMAC of all the above data plus the hour value of current system time in UNIX Epoch time

OllyICE - obfs4proxy.exe - [*.C.P.U* - main thread, module ws2_32]

File View Debug Plugins Options Window Help

Paused

Registers (FPU)

EAX 0000001C
 ECX 00000000
 EDX 116CBB94
 EBX 007777B0 obfs4pro.007777B0
 ESP 0006FE9C
 EBP 0006FEBC
 ESI 116CBBEC
 EDI 0006FEBC
 EIP 764E4406 ws2_32.WSASend

C 0 ES 0023 32bit 0(FFFFFFFF)
 P 1 CS 001B 32bit 0(FFFFFFFF)
 A 0 SS 0023 32bit 0(FFFFFFFF)
 Z 0 DS 0023 32bit 0(FFFFFFFF)
 S 0 FS 003B 32bit 7FFDE000(FFF)

edi=0006FEBC

CALL to WSASend from obfs4pro.0044AC...
 Socket = 124
 Buffers = 116B9AFC
 Buffers = 1
 BytesSent = 116B9AF0
 flags = 0
 Overlapped = 116B9AD0
 Callback = NULL
 RETURN to obfs4pro.004499F4
 obfs4pro.007777B0

representative
 Padding Data
 mark
 HMAC of Entire Data

Obfs4 Client Handshake

Size	Content
20H	Client's representative (Keypair.representative)
variable	Padding data, data size range: 4Dh~1FC0h
10H	mark, HMAC of Client's representative
10H	HMAC of all the above data plus the hour value of current system time in UNIX Epoch time

Registers (FPU)

EAX 0000001C
 ECX 00000000
 EDX 116CBB94
 EBX 007777B0 obfs4pro.007777B0
 ESP 0006FE9C
 EBP 0006FEBC
 ESI 116CBBEC
 EDI 0006FEBC
 EIP 764E4406 ws2_32.WSASend

CALL to WSASend from obfs4pro.0044AC51
 Socket = 124
 Buffers = 116B9AFC
 Buffers = 1
 BytesSent = 116B9AF0
 flags = 0
 Overlapped = 116B9AD0
 Callback = NULL
 RETURN to obfs4pro.004499F4
 obfs4pro.007777B0

representative
 Padding Data
 mark
 HMAC of Entire Data

Obfs4 Client Handshake

Size	Content
20H	Client's representative (Keypair.representative)
variable	Padding data, data size range: 4Dh~1FC0h
10H	mark, HMAC of Client's representative
10H	HMAC of all the above data plus the hour value of current system time in UNIX Epoch time

OllyICE - obfs4proxy.exe - [*C.P.U* - main thread, module ws2_32]

Registers (FPU)

EAX 0000001C
 ECX 00000000
 EDX 116CBB94
 EBX 007777B0 obfs4pro.007777B0
 ESP 0006FE9C
 EBP 0006FEBC
 ESI 116CBBEC
 EDI 0006FEBC
 EIP 764E4406 ws2_32.WSASend

CALL to WSASend from obfs4pro.0044AC...
 Socket = 124
 Buffers = 116B9AFC
 Buffers = 1
 BytesSent = 116B9AF0
 flags = 0
 Overlapped = 116B9AD0
 Callback = NULL
 RETURN to obfs4pro.004499F4
 obfs4pro.007777B0

representative
 Padding Data
 mark
 HMAC of Entire Data

Obfs4 Client Handshake

Size	Content
20H	Client's representative (Keypair.representative)
variable	Padding data, data size range: 4Dh~1FC0h
10H	mark, HMAC of Client's representative
10H	HMAC of all the above data plus the hour value of current system time in UNIX Epoch time

OllyICE - obfs4proxy.exe - [C.P.U* - main thread, module ws2_32]

Registers (FPU)

EAX	0000001C
ECX	00000000
EDX	16CBB94
EBX	07777B0 obfs4pro.007777B0
ESP	006FE9C
EBP	006FEBC
ESI	16CBBEC
EDI	006FEBC
EIP	64E4406 ws2_32.WSASend

Assembly Code:

```

764E4406 8BFF      mov     edi, edi      wsasend
764E4408 55        push   ebp
764E4409 8BEC      mov     ebp, esp
764E440B 51        push   ecx
764E440C 51        push   ecx
764E440D 813D 48705076  cmp    dword ptr [76507048], 764E2E29
764E4417 56        push   esi
764E4418 0F85 CA010000  jnz    764E45E8
764E441E 833D 70705076  cmp    dword ptr [76507070], 0
764E4425 0F84 BD010000  je     764E45E8
764E442B FF35 44705076  push  dword ptr [76507044]
764E4431 FF15 48124E76  call  dword ptr [&API-MS-Win-Core-Prkernel32.TlsGetValue
764E4437 8945 E8      mov     dword ptr [ebp-8], eax
edi=0006FEBC

116FC990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
116FC9A0 55 64 9E 94 6B 93 7F 13 BA 9C A6 E7 02 52 A5 23 03 00 k?!?過?~R?
116FC9B0 D9 90 36 05 65 63 DF 2C 10 01 04 83 0B 77 C9 10 00 00 賽6[ec?+ ]?w?
116FC9C0 B8 A1 42 66 0A 45 FD C6 A9 05 D4 9B 4A D7 F7 1D 浮Bf.E.?語]作
116FC9D0 28 28 76 42 A1 65 2D DF 44 B4 AD 05 F6 03 EF 5D ((vB.-一週喘]?臚
116FC9E0 98 66 CC 29 2E 7A 88 A7 97 FD A8 7E 2C C7 94 75 綴?.z垂胡.善u
116FC9F0 01 AB A8 0F BF 0B AA 4 74 D0 31 CD 55 3C 84 DE .?挑t?胡<勸
116FCA00 AF 59 4D 26 BF 2C E0 BC 6D DD BF 46 0E 51 2F 68 痾M&?嗒m松F的/h
116FCA10 FE 7E E2 AC B4 56 55 3E 22 57 CA 18 92 26 C7 76 秀猜嫩U>?w??夾
116FCA20 66 A6 4F 50 5F 6B 7F D7 6D AB 25 AD 54 F5 2B 36 f.P.k 讚?壘?6
116FCA30 E2 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 割.....
116FCA40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
116FCA50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
116FCA60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
    
```

Registers (FPU) (continued):

C 0	ES	0023	32bit	0(FFFFFFFF)
P 1	CS	001B	32bit	0(FFFFFFFF)
A 0	SS	0023	32bit	0(FFFFFFFF)
Z 0	DS	0023	32bit	0(FFFFFFFF)
S 0	FS	003B	32bit	7FFDE000(FFF)

CALL to WSASend from obfs4pro.0044AC51

representative

Padding Data

mark

HMAC of Entire Data

Obfs4 Client Handshake

Size	Content
20H	Client's representative (Keypair.representative)
variable	Padding data, data size range: 4Dh~1FC0h
10H	mark, HMAC of Client's representative
10H	HMAC of all the above data plus the hour value of current system time in UNIX Epoch time

OllyICE - obfs4proxy.exe - [C.P.U* - main thread, module ws2_32]

Registers (FPU)

EAX 0000001C
 ECX 00000000
 EDX 116CBB94
 EBX 007777B0 obfs4pro.007777B0
 ESP 0006FE9C
 EBP 0006FEBC
 ESI 116CBBEC
 EDI 0006FEBC
 EIP 764E4406 ws2_32.WSASend

CALL to WSARecv from obfs4pro.0044AC...
 Socket = 124
 Buffers = 116B9AFC
 Buffers = 1
 BytesSent = 116B9AF0
 flags = 0
 Overlapped = 116B9AD0
 Callback = NULL
 RETURN to obfs4pro.004499F4
 obfs4pro.007777B0

representative
 Padding Data
 mark
 HMAC of Entire Data

Server Verifies Client's Handshake

- Generate server's own KeyPair instance
- Verify client's handshake and restore client's public key
- **ECC Scalar Multiplication** - `curve25519.ScalarMult` (server's private key, client's public key)

The Very Important Part of ECC Algorithm:

```
ScalarMult(server.privateKey, client.publicKey) == ScalarMult(client.privateKey, server.publicKey)
```

On Server Side

On Client Side

Obfs4 Server's Handshake

Size	Content
20H	Server's representative (Keypair.representative)
20H	Server's auth
variable	Padding data, size range: 0h~1F73h
10H	mark, HMAC of Server's representative
10H	HMAC of all the above data plus the hour value of current system time in UNIX Epoch time

- Similar to client's handshake packet
- Server's auth is added for client **authentication**
- Use a different padding data size range

Generate Common keySeed and Verify Server's Auth

- Client calls **curve25519.ScalarMult** (client's private key, server's public key) and **curve25519.ScalarMult** (client's private key , server's id public key)
- Generate the common **keySeed** with above two function results and verify server's auth.
- **Final Encryption/Decryption keys** are generated based on the common keySeed

Obfs4 Seals/Unseals Tor Payload

- **Write()**
 - makePacket()
 - **Encrypt (encode)** Tor Payload (conn.encoder.Encode())
 - Append random padding to encrypted payload
 - IAT-Mode
- **Read()**
 - readPackets()
 - **Decrypt (decode)** Tor Payload (conn.decoder.Decode())

IAT Mode

- MTU (Maximum Transmission Unit)
- Network device splits large size packets into MTU size packets, which can be easily reassembled and identified
- IAT (Inter-Arrival Timing) mode
- The value can be 0, 1 and 2
 - 0 IAT mode disabled for this bridge relay
 - 1 split into MTU size packets, **1448 bytes**
 - 2 split into **variable** size packets

```
60 //
61 package framing // import "gitlab.com/yawning/obfs4.git/transport/obfs4/framing"
62
63 import (
64     "bytes"
65     "encoding/binary"
66     "errors"
67     "fmt"
68     "io"
69
70     "gitlab.com/yawning/obfs4.git/common/crand"
71     "gitlab.com/yawning/obfs4.git/common/drbg"
72     "golang.org/x/crypto/nacl/secretbox"
73 )
74
75 const (
76     // MaximumSegmentLength is the length of the largest possible segment
77     // including overhead.
78     MaximumSegmentLength = 1500 - (40 + 12) //by zxp, 1448. 5a8H.
79
80     // FrameOverhead is the length of the framing overhead.
81     FrameOverhead = lengthLength + secretbox.Overhead
82
83     // MaximumFramePayloadLength is the length of the maximum allowed payload
84     // per frame. MaximumFramePayloadLength=1448-18=1430 i.e. 596h
85     MaximumFramePayloadLength = MaximumSegmentLength - FrameOverhead
86
87     // KeyLength is the length of the Encoder/Decoder secret key.
88     //const SeedLength = 16 + Size// Size = 8, =24 i.e. 18H
```


IAT Mode

tcp.stream eq 0

Time	Source	Destination	Protocol	Info
264	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=4428 Ack=10184 Win=63615 Len=1448
265	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=10184 Ack=5876 Win=65535 Len=0
278	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=5876 Ack=10184 Win=63615 Len=1271
279	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=10184 Ack=7147 Win=65535 Len=0
374	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=10184 Ack=7147 Win=65535 Len=1420
375	38.229.33.83	10.0.2.15	TCP	80 → 63169 [PSH, ACK] Seq=11604 Ack=7147 Win=65535 Len=28
376	10.0.2.15	38.229.33.83	TCP	63169 → 80 [ACK] Seq=7147 Ack=11632 Win=64240 Len=0
380	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=7147 Ack=11632 Win=64240 Len=1448
381	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=11632 Ack=8595 Win=65535 Len=0
382	38.229.33.83	10.0.2.15	TCP	80 → 63169 [PSH, ACK] Seq=11632 Ack=8595 Win=65535 Len=625
386	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=8595 Ack=12257 Win=63615 Len=1448
387	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=10043 Win=65535 Len=0
388	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=10043 Ack=12257 Win=63615 Len=1303
389	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=11346 Win=65535 Len=0
396	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=11346 Ack=12257 Win=63615 Len=1448
397	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=12794 Win=65535 Len=0
398	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=12794 Ack=12257 Win=63615 Len=1448
399	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=14242 Win=65535 Len=0
408	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=14242 Ack=12257 Win=63615 Len=1448
409	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=15690 Win=65535 Len=0
424	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=15690 Ack=12257 Win=63615 Len=1448
428	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=17138 Win=65535 Len=0
433	10.0.2.15	38.229.33.83	TCP	63169 → 80 [PSH, ACK] Seq=17138 Ack=12257 Win=63615 Len=625
434	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=17763 Win=65535 Len=0
449	38.229.33.83	10.0.2.15	TCP	80 → 63169 [ACK] Seq=12257 Ack=17763 Win=65535 Len=1420
450	38.229.33.83	10.0.2.15	TCP	80 → 63169 [PSH, ACK] Seq=13677 Ack=17763 Win=65535 Len=28

Conclusion

Hard to Censor

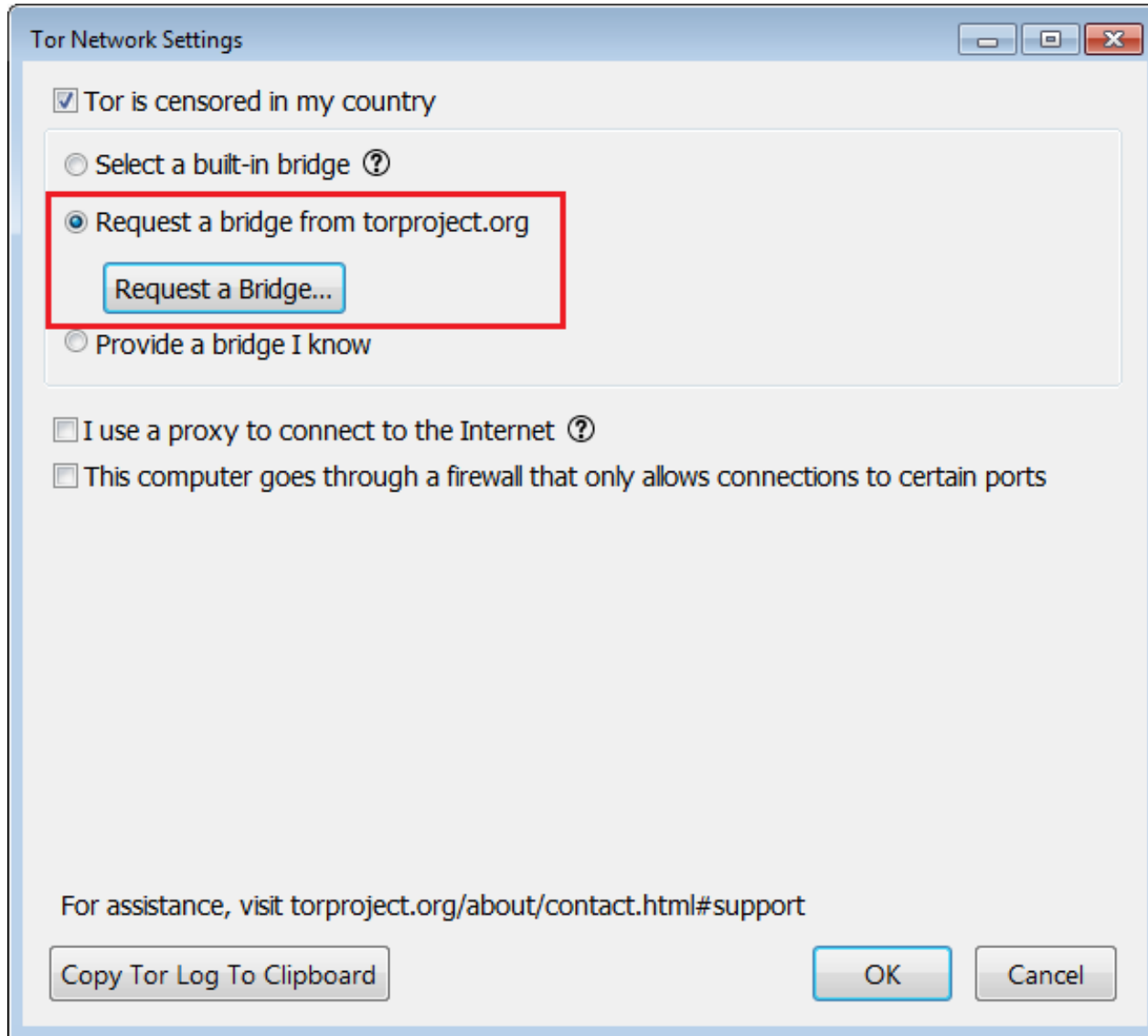
The Tor traffic powered by Obfs4-bridge is harder to be censored because:

- Obfs4 **encrypts** Tor traffic
- Obfs4 **packet size** is obfuscated by adding padding data, even the Handshake packet
- Obfs4 large packet can be split by **IAT mode**
- Besides those built-in Obfs4 Bridges, Tor provides **three other ways** to obtain more private Obfs4 Bridges

Three Ways To Obtain Obfs4 Bridge

- Request through **Tor Network Settings**.

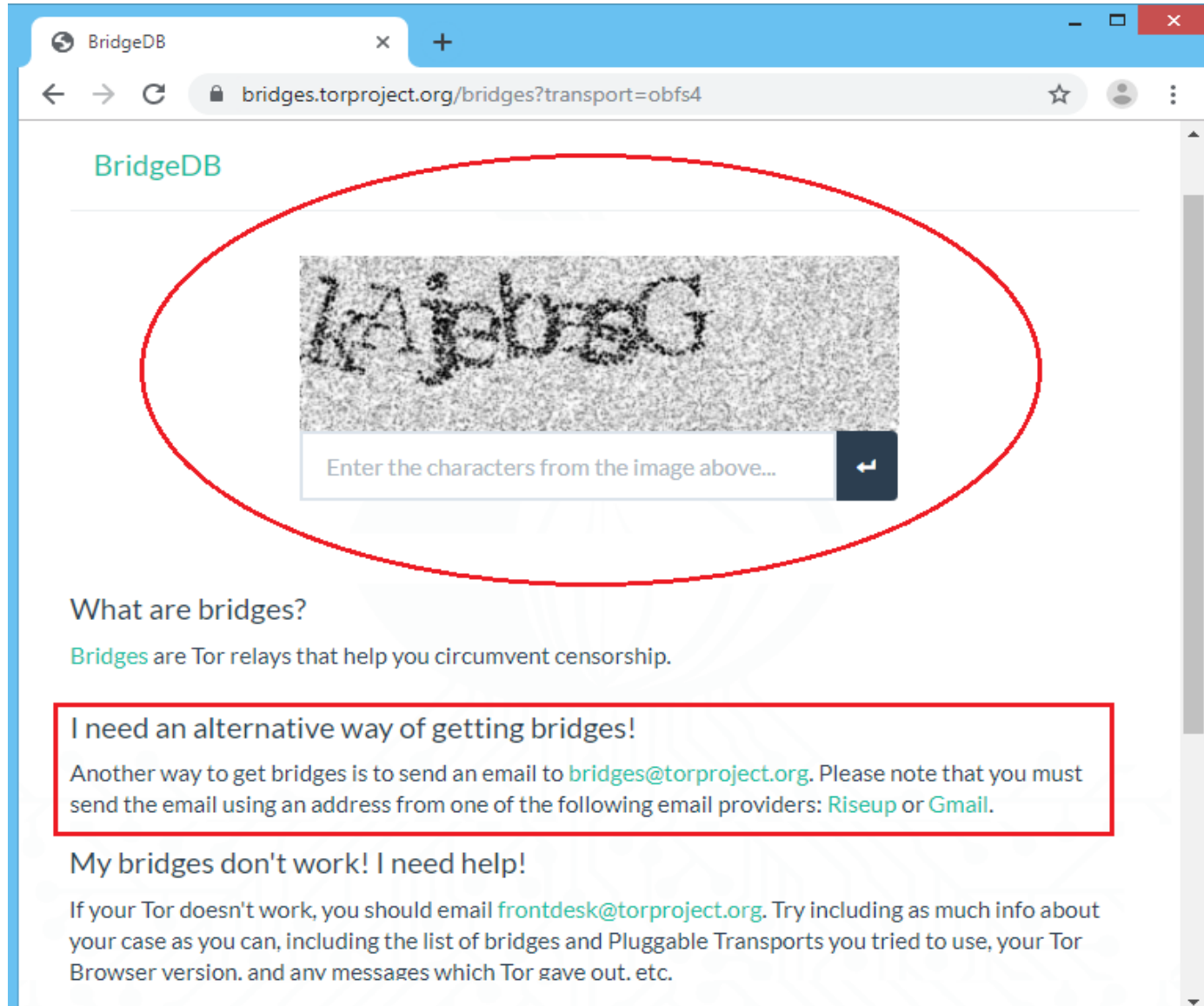
Three Ways To Obtain Obfs4 Bridge



Three Ways To Obtain Obfs4 Bridge

- Request through **Tor Network Settings**.
- Request on **Tor Web Site**.
- Request via **E-Mail**.

Three Ways To Obtain Obfs4 Bridge



The screenshot shows a web browser window with the address bar displaying `bridges.torproject.org/bridges?transport=obfs4`. The page title is "BridgeDB". A CAPTCHA image is displayed, showing the word "Algebra" in a distorted, pixelated font. Below the image is a text input field with the placeholder text "Enter the characters from the image above..." and a submit button with a right-pointing arrow. A red oval highlights the CAPTCHA image and the input field.

BridgeDB

Enter the characters from the image above...

What are bridges?
Bridges are Tor relays that help you circumvent censorship.

I need an alternative way of getting bridges!
Another way to get bridges is to send an email to bridges@torproject.org. Please note that you must send the email using an address from one of the following email providers: [Riseup](#) or [Gmail](#).

My bridges don't work! I need help!
If your Tor doesn't work, you should email frontdesk@torproject.org. Try including as much info about your case as you can, including the list of bridges and Pluggable Transports you tried to use, your Tor Browser version, and any messages which Tor gave out, etc.

Three Ways To Obtain Obfs4 Bridge

The image shows a composite screenshot illustrating the process of obtaining an Obfs4 bridge. On the left, the 'Tor Network Settings' window is open, with the 'Provide a bridge I know' option selected. A text input field is present with the placeholder text 'type address:port (one per line)'. A red arrow points from the 'Copy & Paste' text to this input field. On the right, a browser window shows the BridgeDB website at <https://bridges.torproject.org/bridges?transport=obfs4>. The page displays the heading 'Here are your bridge lines:' followed by a list of three bridge lines:

```
obfs4 139.162.51.145:9080 3D2A926257DEAE24D7572C399FA1253357E9538B  
obfs4 185.163.45.86:443 AE7DCAA2559637FA216EFA7BAB2416A0527C8D23 c  
obfs4 195.123.245.176:30384 82468406637273DC32D1AB8076D7F1D89B6A19:
```

At the bottom of the browser window, there are buttons for 'Select All' and 'Show QRCode'.

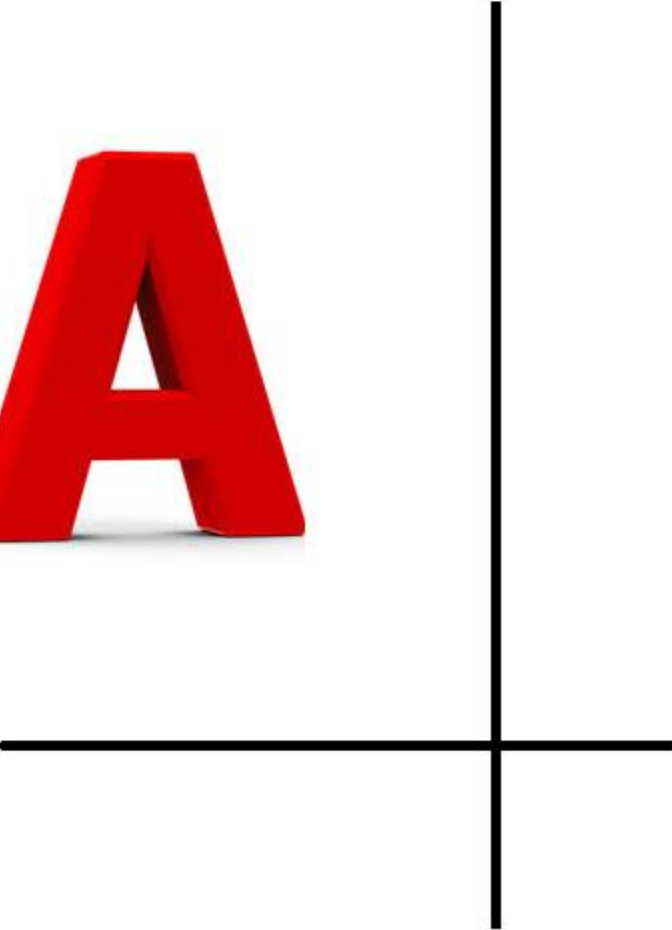
References

- <https://www.torproject.org/>
- [https://en.wikipedia.org/wiki/Tor_\(anonymity_network\)](https://en.wikipedia.org/wiki/Tor_(anonymity_network))
- <https://github.com/Yawning/obfs4>
- <https://blog.torproject.org/tor-heart-bridges-and-pluggable-transport>
- <https://bridges.torproject.org/bridges?transport=obfs4>
- https://en.wikipedia.org/wiki/Onion_routing

Tor



Q&A



Thank You!