MTAT.07.017
Applied Cryptography

The Onion Router (Tor)

University of Tartu

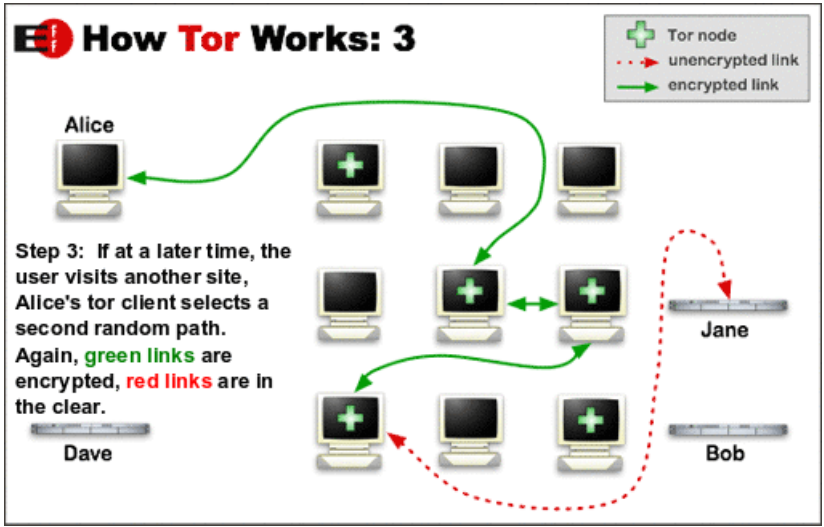
Spring 2020

Tor

"Tor is a software for enabling online anonymity and censorship resistance. Tor directs Internet traffic through a free, worldwide, volunteer network consisting of more than seven thousand relays to conceal a user's location or usage from anyone conducting network surveillance or traffic analysis."

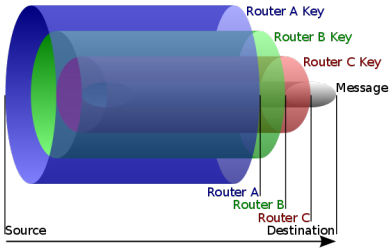
`https://en.wikipedia.org/wiki/Tor_(anonymity_network)`

Tor

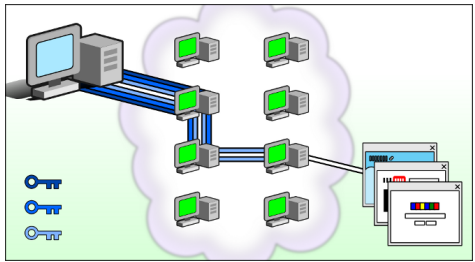


- No single node knows the entire path!

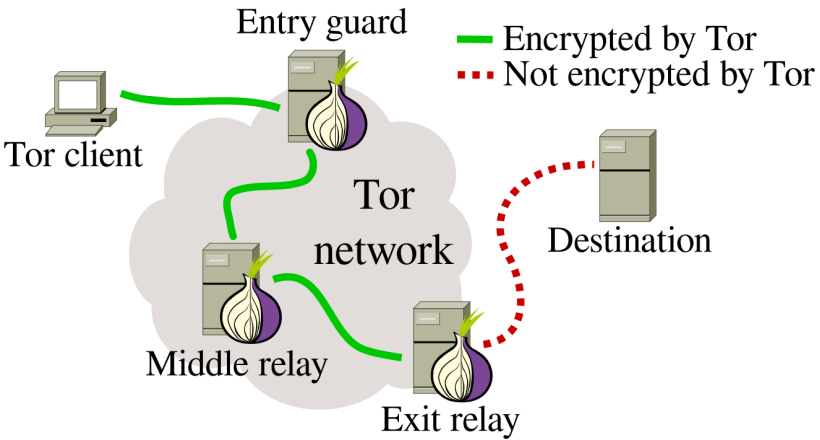
Onion Routing



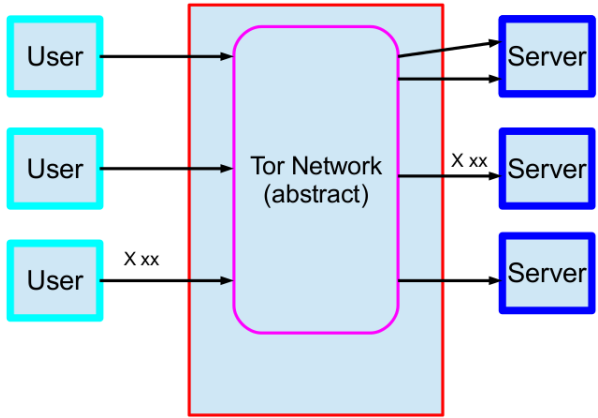
- Data received by router A:
  - $E_a(b, E_b(c, E_c(dest, m)))$
- Data received by router B:
  - $E_b(c, E_c(dest, m))$
- Data received by router C:
  - $E_c(dest, m)$



Tor Nodes

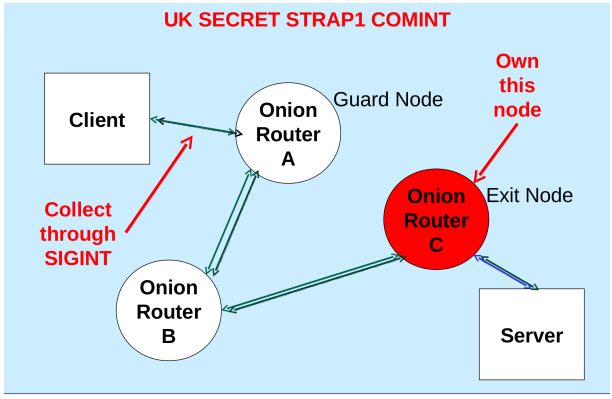

Entry guard
Encrypted by Tor
Not encrypted by Tor
Tor client
Tor
network
Destination
Middle relay
Exit relay

Traffic correlation attack



- End-to-end correlation
- No need to know the full path
- Entry guard should not be rotated frequently

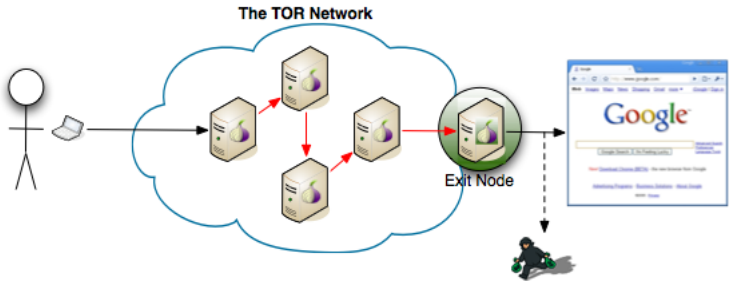
GCHQ attacks on Tor



https://archive.org/details/spiegel_-_media-35538

- How many Tor nodes are run by GCHQ/NSA?
- Tor nodes: `https://torstatus.rueckgr.at/`

Malicious exit node attacks



- Sniffing
  - WikiLeaks
- Man-in-the-middle attacks
  - “Spoiled Onions: Exposing Malicious Tor Exit Relays”

Malicious exit node attacks

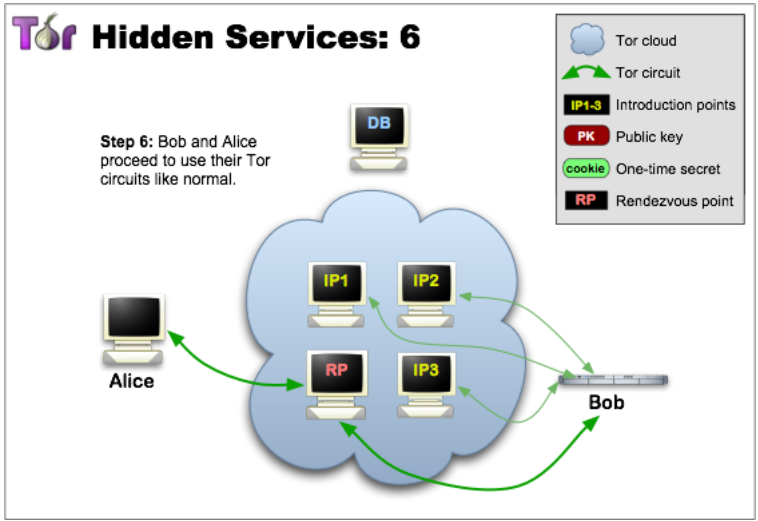
Fingerprint	IP addresses	Country	Bandwidth	Attack	Sampling rate	First active	Discovery
F8FD29D0†	176.99.12.246	Russia	7.16 MB/s	HTTPS MitM	*unknown*	2013-06-24	2013-07-13
8F9121BF†	64.22.111.168/29	U.S.	7.16 MB/s	HTTPS MitM	*unknown*	2013-06-11	2013-07-13
93213A1F†	176.99.9.114	Russia	290 KB/s	HTTPS MitM	50%	2013-07-23	2013-09-19
05AD06E2†	92.63.102.68	Russia	5.55 MB/s	HTTPS MitM	33%	2013-08-01	2013-09-19
45C55E46†	46.254.19.140	Russia	1.54 MB/s	SSH & HTTPS MitM	12%	2013-08-09	2013-09-23
CA1BA219†	176.99.9.111	Russia	334 KB/s	HTTPS MitM	37.5%	2013-09-26	2013-10-01
1D70CDED†	46.38.50.54	Russia	929 KB/s	HTTPS MitM	50%	2013-09-27	2013-10-14
EE215500†	31.41.45.235	Russia	2.96 MB/s	HTTPS MitM	50%	2013-09-26	2013-10-15
12459837†	195.2.252.117	Russia	3.45 MB/s	HTTPS MitM	26.9%	2013-09-26	2013-10-16
B5906553†	83.172.8.4	Russia	850.9 KB/s	HTTPS MitM	68%	2013-08-12	2013-10-16
EFF1D805†	188.120.228.103	Russia	287.6 KB/s	HTTPS MitM	61.2%	2013-10-23	2013-10-23
229C3722	121.54.175.51	Hong Kong	106.4 KB/s	sslstrip	*unsampled*	2013-06-05	2013-10-31
4E8401D7†	176.99.11.182	Russia	1.54 MB/s	HTTPS MitM	79.6%	2013-11-08	2013-11-09
27FB6BB0†	195.2.253.159	Russia	721 KB/s	HTTPS MitM	43.8%	2013-11-08	2013-11-09
0ABB31BD†	195.88.208.137	Russia	2.3 MB/s	SSH & HTTPS MitM	85.7%	2013-10-31	2013-11-21
CADA00B9†	5.63.154.230	Russia	187.62 KB/s	HTTPS MitM	*unsampled*	2013-11-26	2013-11-26
C1C0EDAD†	93.170.130.194	Russia	838.54 KB/s	HTTPS MitM	*unsampled*	2013-11-26	2013-11-27
5A2A51D4	111.240.0.0/12	Taiwan	192.54 KB/s	HTML Injection	*unsampled*	2013-11-23	2013-11-27
EBF7172E†	37.143.11.220	Russia	4.34 MB/s	SSH MitM	*unsampled*	2013-11-15	2013-11-27
68E682DF†	46.17.46.108	Russia	60.21 KB/s	SSH & HTTPS MitM	*unsampled*	2013-12-02	2013-12-02
533FDE2F†	62.109.22.20	Russia	896.42 KB/s	SSH & HTTPS MitM	42.1%	2013-12-06	2013-12-08
E455A115	89.128.56.73	Spain	54.27 KB/s	sslstrip	*unsampled*	2013-12-17	2013-12-18
02013F48	117.18.118.136	Hong Kong	538.45 KB/s	DNS censorship	*unsampled*	2013-12-22	2014-01-01
2F5B07B2	178.211.39	Turkey	204.8 KB/s	DNS censorship	*unsampled*	2013-12-28	2014-01-06
4E2692FE	24.84.118.132	Canada	52.22 KB/s	OpenDNS	*unsampled*	2013-12-21	2014-01-06

Deanonymization attacks

- Browser fingerprinting
- IP leakage through DNS requests
- Client-side exploits
- Tor Browser Bundle

Tor Onion Services

Reachable using .onion address (e.g., `duskgytldkxiuqc6.onion`)

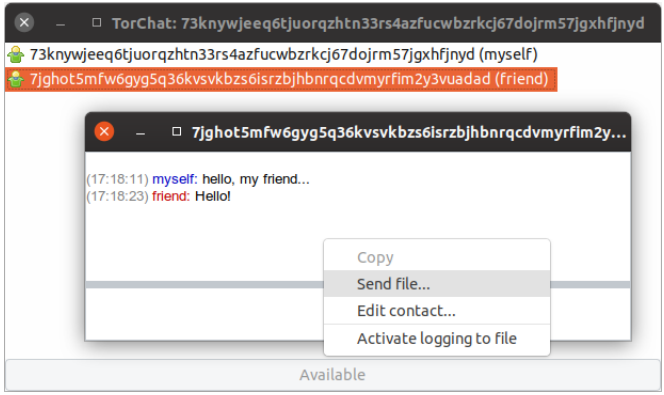


Tor Onion Services (setup)

```
# apt install tor

# cat /etc/tor/torrc | grep HiddenService
HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 127.0.0.1:80

# /etc/init.d/tor restart

# hexdump -C /var/lib/tor/hidden_service/hs_ed25519_secret_key
00000000  3d 3d 20 65 64 32 35 35  31 39 76 31 2d 73 65 63  |== ed25519v1-sec|
00000010  72 65 74 3a 20 74 79 70  65 30 20 3d 3d 00 00 00  |ret: type0 ==...|
00000020  90 e5 5b 2b c4 77 9a 80  bf 2f 63 90 a7 1d e2 73  |..[+.w.../c....s|
...

# cat /var/lib/tor/hidden_service/hostname
7jghot5mfw6gyg5q36kvsvkbzs6isrzbjhbnrqcdvmyrfim2y3vuadad.onion
```

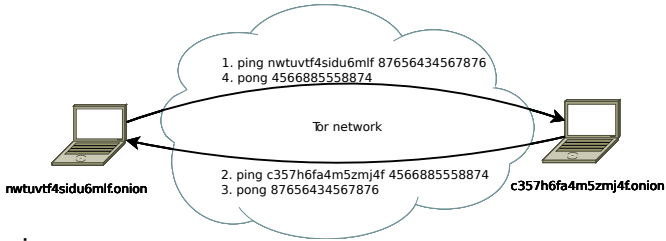
TorChat (Instant Messenger)

```
$ sudo dpkg -i torchat.deb
$ torchat
```



http://kodu.ut.ee/~arnis/torchat_thesis.pdf

TorChat Protocol

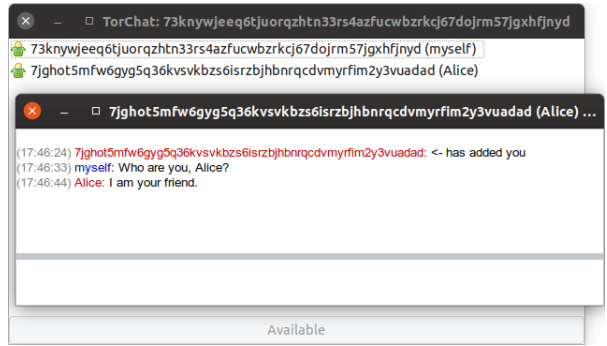


- Two connections:
  - Outgoing connection to send data
  - Incoming connection to receive data
- The handshake process authenticates the incoming connections
- Commands (command separator is the newline `\n` character):
  - `message` – UTF-8 encoded message
  - `status` – `available`, `away` or `xa`
  - `add_me` – request addition to buddy list
  - `profile_name` – name in the buddy list

Task: TorChat – 5p

Implement a TorChat client that is compatible with the official TorChat client:

```
$ ./torchat.py --myself 7jghot5mfw6gyg5q36kvsvkbzs6isrzbjhbnrqcdvmyrfim2y3vuadad --peer 73knywjeeq6tjuorqzhtn33rs4a
[+] Connecting to peer 73knywjeeq6tjuorqzhtn33rs4azfucwbzrkcj67dojrm57jgxhfjnyd
[+] Sending: ping 7jghot5mfw6gyg5q36kvsvkbzs6isrzbjhbnrqcdvmyrfim2y3vuadad 362515936879510133458293327387504010 89
[+] Listening...
[+] Client 127.0.0.1:49900
[+] Received: ping 73knywjeeq6tjuorqzhtn33rs4azfucwbzrkcj67dojrm57jgxhfjnyd 114550899558407750467319526697617679065
[+] Received: pong 362515936879510133458293327387504010 89
[+] Incoming connection authenticated!
[+] Sending: pong 114550899558407750467319526697617679065735261062805617012379414337324471905588
[+] Received: client TorChat
[+] Received: version 0.9.9.553
[+] Received: profile_name Bob
[+] Received: status available
[+] Sending: add_me
[+] Sending: status available
[+] Sending: profile_name Alice
[+] Received: status available
[+] Received: message Who are you, Alice?
[?] Enter message: I am your friend.
[+] Sending: message I am your friend.
[+] Received: status available
[+] Received: status available
[+] Received: status available
```



Task: TorChat

- Setup a Tor onion service for port 11009 (redirected to 127.0.0.1:8888)
- Connect to peer's `.onion` address on port 11009 and send the `ping` command with a random cookie
  - The cookie must be a random decimal number with 128-bit entropy
- Listen on 127.0.0.1:8888 for peer's connect back
  - Verify that the TorChat ID in the peer's `ping` command is correct
  - Verify that the cookie in the peer's `pong` command matches the cookie you sent
- Send the `pong` command with peer's cookie over the outgoing connection (only after the `pong` has been verified)
- Send the `add_me`, `status` and `profile_name` commands
- After every `message` received, read the user input and send the response `message` (only after the incoming connection has been authenticated)

Server sockets in Python

```
import socket
sserv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sserv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sserv.bind(('', 8888))
sserv.listen(0)

(s, address) = sserv.accept()
print("[+] Client %s:%s" % (address[0], address[1]))
```

- `bind(('', 8888))` and `listen()` listens for client connections on all IPs on all network interfaces
- `accept()` will wait until client connects and will return a tuple:
  - client socket (has `send()` and `recv()` methods)
  - address tuple – IP and port
- `SO_REUSEADDR` forces the kernel to reuse port even if it is in busy (`TIME_WAIT`) state (prevents error when rebinding)

`http://docs.python.org/3/howto/sockets.html`

Most common pitfalls

- Cannot connect to the peer
  - Check that Tor service is running
  - Check that the official TorChat is running and its status is online
  - Check that the ".onion" suffix in the hostname is not missing
- Connect-back from the peer not received
  - Check that your TorChat ID in `ping` command is correct
  - Check that your `pong` command ends with the command separator
  - To test whether your onion service is available over the Tor, use: "`torify telnet youronionaddress.onion 11009`"
- The peer ignores your commands
  - Check that your commands are sent over the correct socket
  - Check that what you print is what you send

Questions

- What is the security objective Tor tries to achieve?
- Tor middle node sees only encrypted packets. How it is achieved?
- What could a malicious Tor exit node do?
- What could a malicious Tor middle node do?
- What could a malicious Tor entry guard node do?
- How to detect whether the user is using Tor network?
- Under what threat model Tor is secure?
- How are Tor Onion Services identified?
- What prevents someone from impersonating Tor Onion Service?