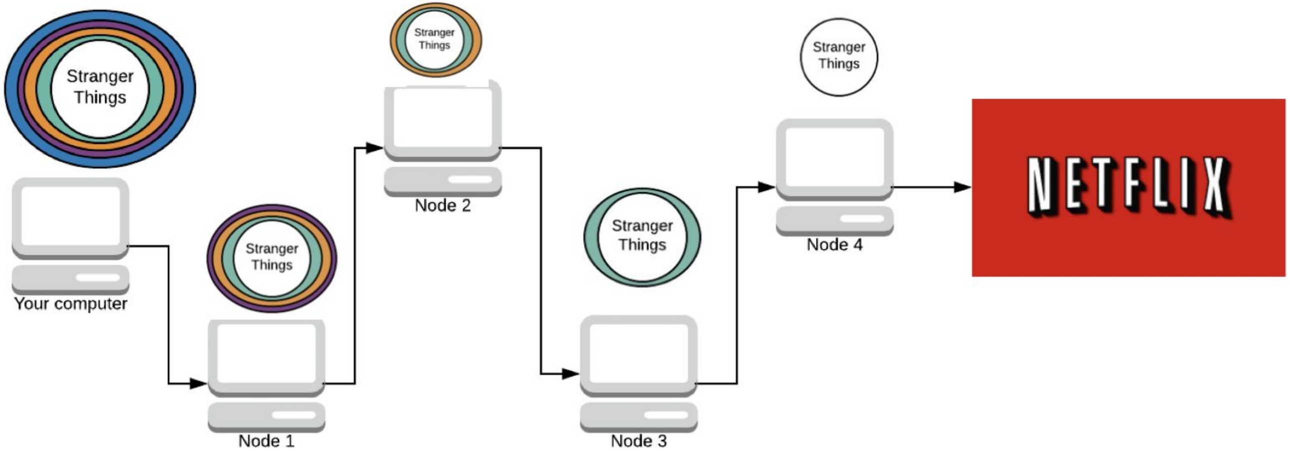
Tor@CORE

Zhuoyu Ji / **Rong** Jin / **Haoyang** Wang / **Peiyang** Yu

What is Tor?

- The Onion Routing
  - Circuit with multiple nodes instead of direct connection
  - Multiple layers of encryption
  - Peeling off/wrapping up encrypted message at each onion router



What is Tor? (cont.)

- Goal: **Secure** and **Anonymous** communication
  - Does not provide complete security or anonymity
  - Secure as long as entry & exit relays are not both compromised
  - Anonymity comes from lack of full information

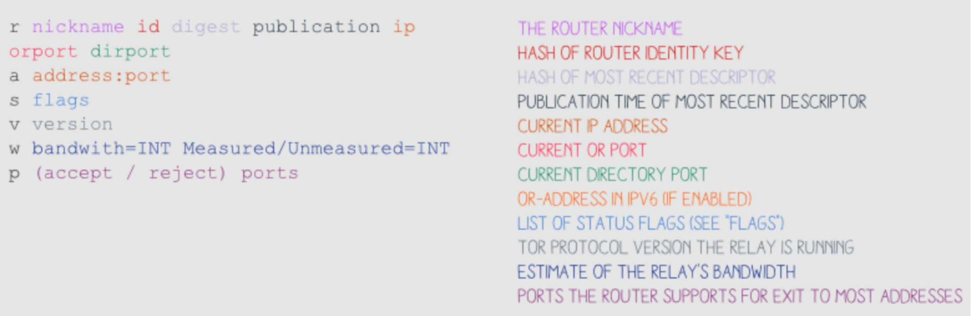
- "Entry to the dark web" :)
- First developed by US Naval Research Laboratory
  (Yes, the same group that developed CORE!)

Project Overview

- Resources:
    - Official implementation
    - Official documentation and specification
    - Existing custom implementations (Torpy, TinyTor)
- Goals:
    - Build a naive Tor model in CORE
        - Approximate the basic Tor functionality
        - Simplify security parts (authentication, encryption)
    - Test the performance of our model
        - Compare with direct connection
        - Compare different node number

Tor Component: Relay

- Our design:
  - Send simple **control message** to Directory Authorities (DA)
  - **Peel one layer of onion off** while relaying client request to server
  - **Add one layer of onion on** while relaying server response to client
- Simplification from real world Tor:
  - Real world Tor Relay has **multiple public/private key pairs**
    - 3 1024-bit RSA keys, 1 Curve25519 Key, 3 Ed25519 Keys
  - Real world Tor Relay sends **much more metadata** about itself to the DA

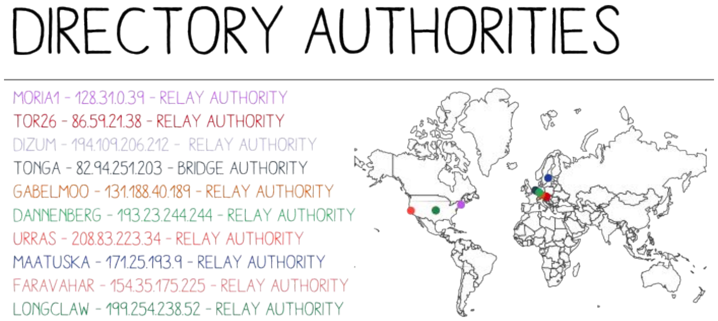


```
r nickname id digest publication ip
orport dirport
a address:port
s flags
v version
w bandwith=INT Measured/Unmeasured=INT
p (accept / reject) ports
```

THE ROUTER NICKNAME
HASH OF ROUTER IDENTITY KEY
HASH OF MOST RECENT DESCRIPTOR
PUBLICATION TIME OF MOST RECENT DESCRIPTOR
CURRENT IP ADDRESS
CURRENT OR PORT
CURRENT DIRECTORY PORT
OR-ADDRESS IN IPV6 (IF ENABLED)
LIST OF STATUS FLAGS (SEE "FLAGS")
TOR PROTOCOL VERSION THE RELAY IS RUNNING
ESTIMATE OF THE RELAY'S BANDWIDTH
PORTS THE ROUTER SUPPORTS FOR EXIT TO MOST ADDRESSES

Tor Component: Directory Authority

- A few servers to track state of the entire Tor network
  - Redundancy and distributed trust
- Our design:
  - **No communication** between DAs
  - Maintain metadata for relay nodes
  - Send relay info to client **including keys**
- Simplification from real world Tor:
  - Real world DA is a distributed system with authoritative directories and their mirrors
  - Real world DA communicates to form a **consensus on relay information**
  - Real world DA involves in client-relay key exchange



DIRECTORY AUTHORITIES

MORIA1 - 128.31.0.39 - RELAY AUTHORITY

TOR26 - 86.59.21.38 - RELAY AUTHORITY

DIZUM - 194.109.206.212 - RELAY AUTHORITY

TONGA - 82.94.251.203 - BRIDGE AUTHORITY

GABELMOO - 131.188.40.189 - RELAY AUTHORITY

DANNENBERG - 193.23.244.244 - RELAY AUTHORITY

URRAS - 208.83.223.34 - RELAY AUTHORITY

MAATUSKA - 171.25.193.9 - RELAY AUTHORITY

FARAVAHAR - 154.35.175.225 - RELAY AUTHORITY

LONGCLAW - 199.254.238.52 - RELAY AUTHORITY

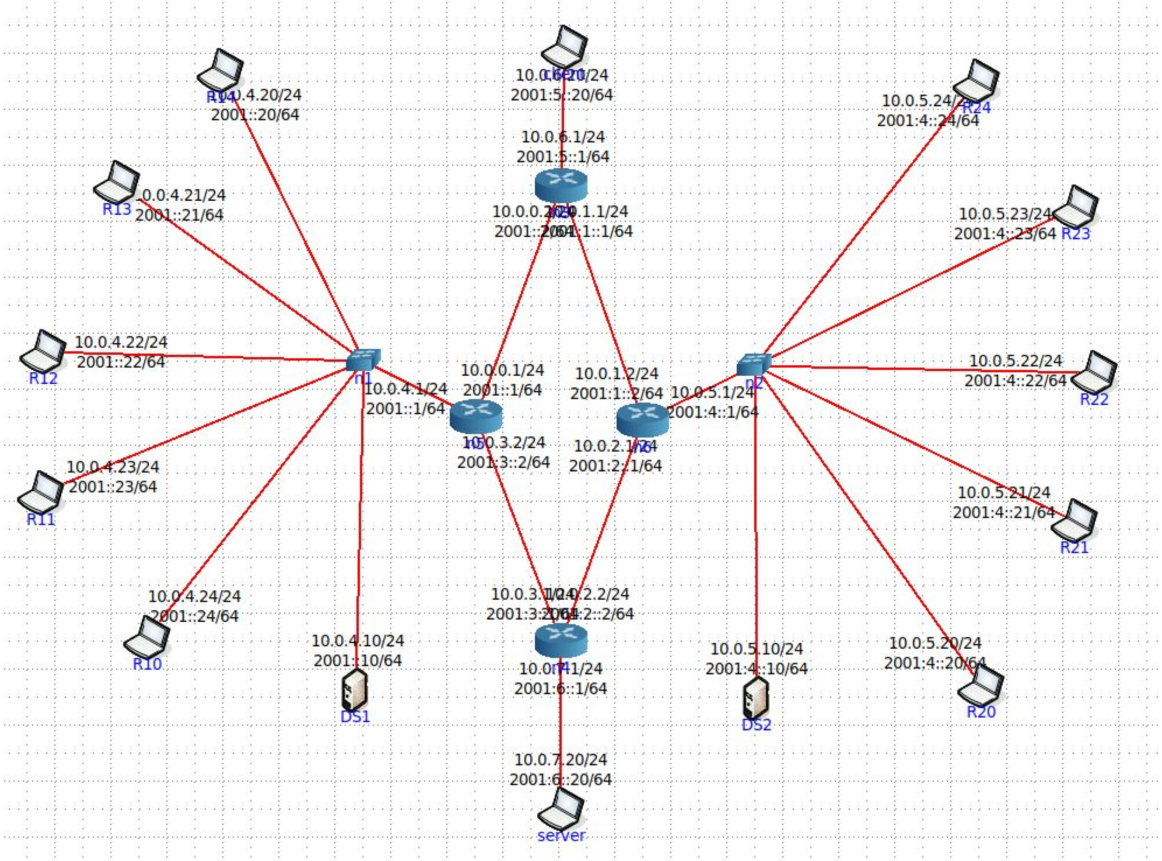
Tor Component: Client

- User Interface to visit website using Tor network
- Our design:
  - Communicate with Directory Authority to get relay information
  - **Unify information from DA** and select circuit straightforwardly
- Simplification from real world Tor:
  - Real world Tor client uses **AES with Diffie-Hellman**, we simply use **Caesar Cipher** to simplify encryption
  - Real world Tor does circuit selection in a more sophisticated way

Other Simplifications We Made

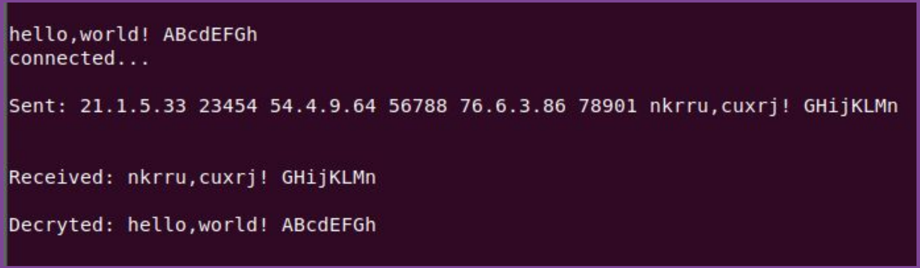
- Circuit selection restriction
- Tor Bridge
- Packet format and restriction
- Security components:
    - Key exchange
    - Encryption algorithm

Putting Components Together



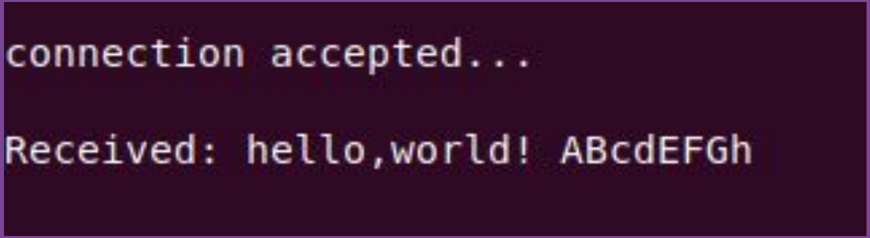
Tor@CORE in Action

Client



```
hello,world! ABcdEFGh
connected...

Sent: 21.1.5.33 23454 54.4.9.64 56788 76.6.3.86 78901 nkrru,cuxrj! GHijKLMn


Received: nkrru,cuxrj! GHijKLMn

Decryted: hello,world! ABcdEFGh
```

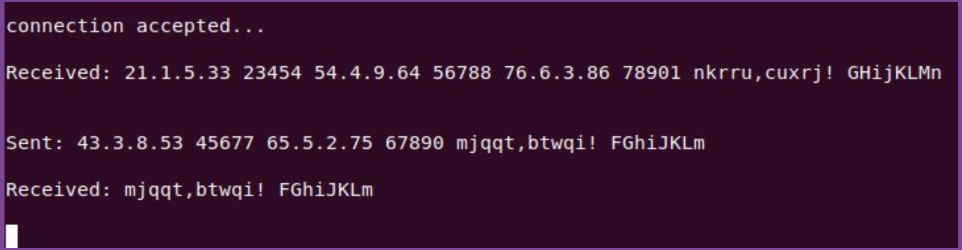
Server



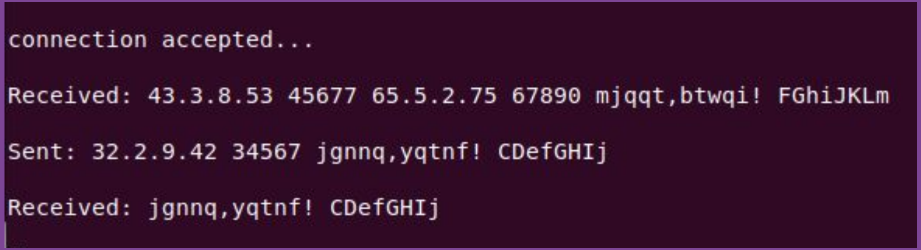
```
connection accepted...

Received: hello,world! ABcdEFGh
```

request direction

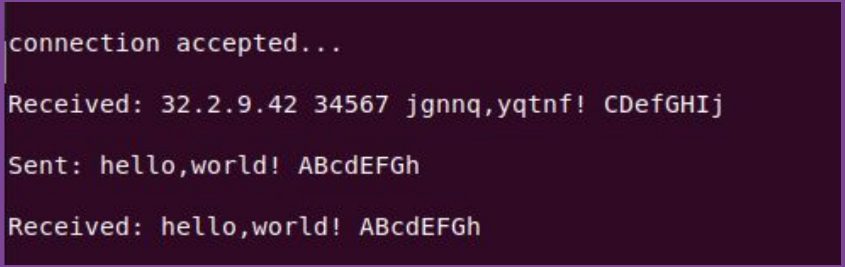
Three Relay Nodes



```
connection accepted...

Received: 21.1.5.33 23454 54.4.9.64 56788 76.6.3.86 78901 nkrru,cuxrj! GHijKLMn


Sent: 43.3.8.53 45677 65.5.2.75 67890 mjqqt,btwqi! FGhiJKLm

Received: mjqqt,btwqi! FGhiJKLm
```



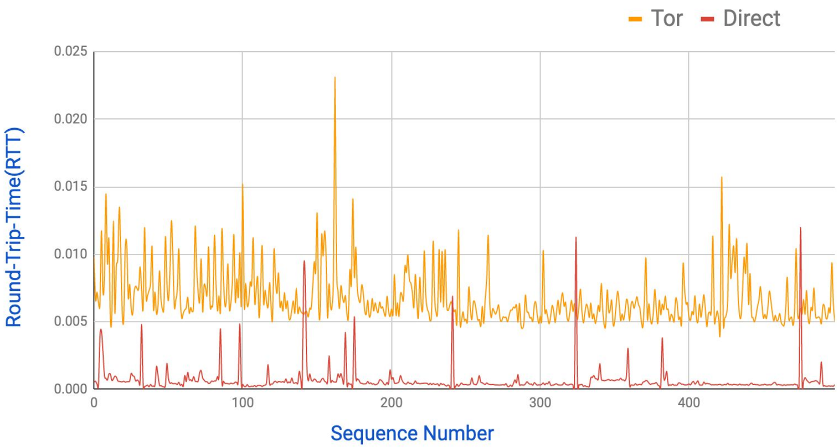
```
connection accepted...

Received: 43.3.8.53 45677 65.5.2.75 67890 mjqqt,btwqi! FGhiJKLm

Sent: 32.2.9.42 34567 jgnnq,yqtnf! CDefGHIj

Received: jgnnq,yqtnf! CDefGHIj
```

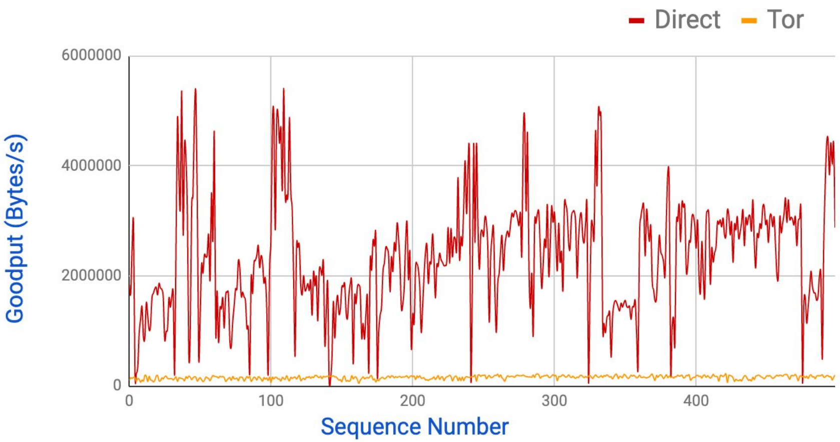


```
connection accepted...

Received: 32.2.9.42 34567 jgnnq,yqtnf! CDefGHIj

Sent: hello,world! ABcdEFGh

Received: hello,world! ABcdEFGh
```

Performance: Direct vs 3-Node Tor

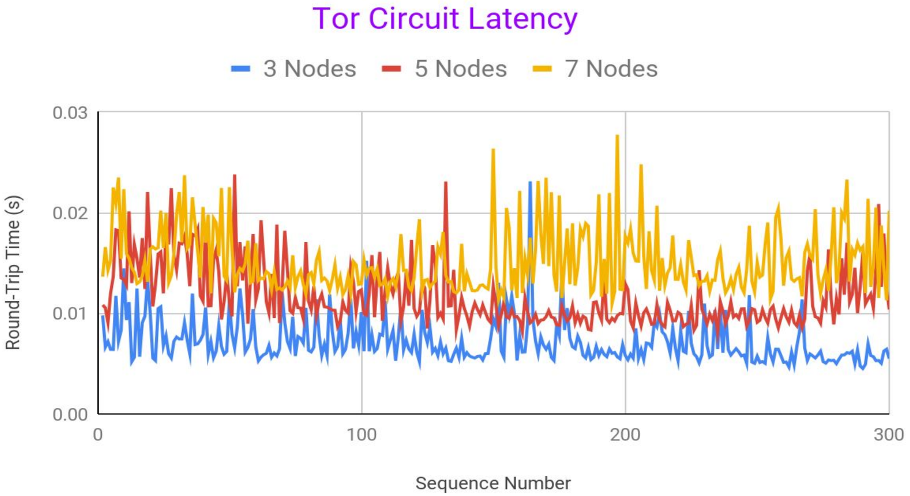
	Direct	3-Node Tor
Average RTT (s)	0.0006	0.0062
Average Goodput (B/s)	2444615	167702

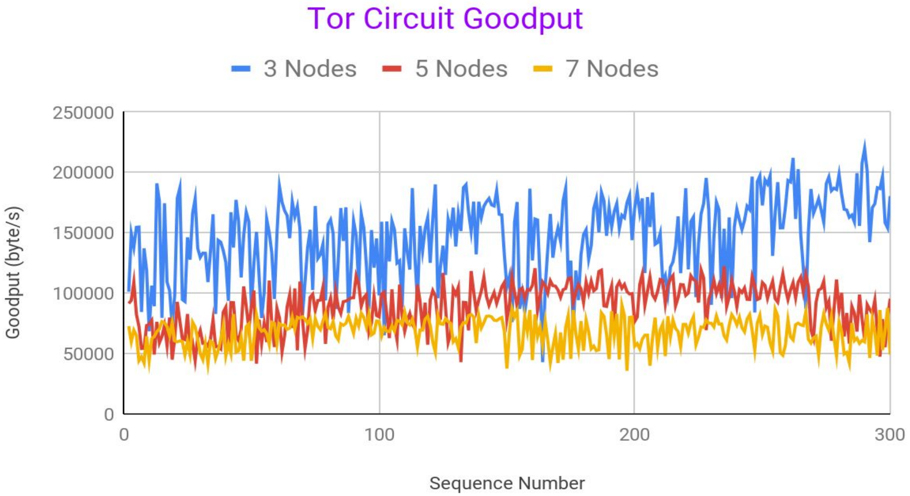

RTT: Direct vs 3-Nodes Tor


Goodput: Direct vs 3-Nodes Tor

Performance: Tor with different Node

	3-Node	5-Node	7-Node
Average RTT (s)	0.0062	0.0102	0.0143
Average Goodput (B/s)	167702	100636	70629





Conclusion

- Tor is really slow!
- Without our simplifications, real world Tor should be much slower
- 3 Relay Node is optimal:
    - Adding relays does not increase security or anonymity
    - Adding each relay adds decent amount of latency (~0.004s in our model)

Future Work:

- Implement some parts we simplified
- Test in a more realistic scenario
- Try to interact with real-world Tor

Q & A