The Onion Name System: Tor-Powered Distributed DNS for Tor Hidden Services

Jesse Victors
Network Security

Introduction - TCP/IP

- Fundamental Internet communication
- Header
  - Routing information
  - Connection details
- Body
  - Contents of messages
- Leaks connection and messages

Introduction - SSL & TLS

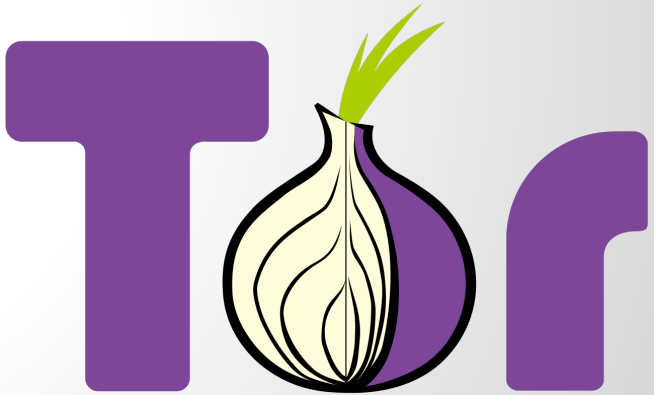
- Set of protocols to establish a private communication channel
- End-to-end encryption between both parties
- Does not encrypt TCP header
  - Necessary for routing
  - Leaks who you are communicating to
  - Sufficient to break privacy

Introduction - Privacy Systems

- Family of tools
- Obscures link between user identity and activities
- Obfuscating traffic patterns
- Resistance to traffic analysis
- Censorship resistance
- Most descend from mixnets (1981)
- High latency or low latency
  - High latency defends against a global attacker

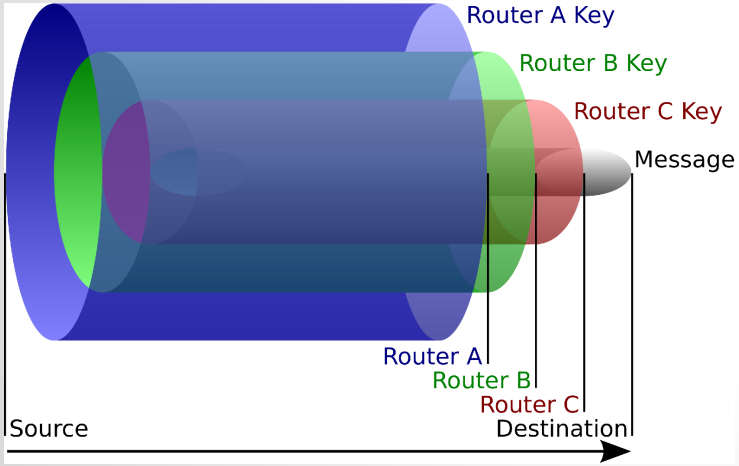
Background - Tor

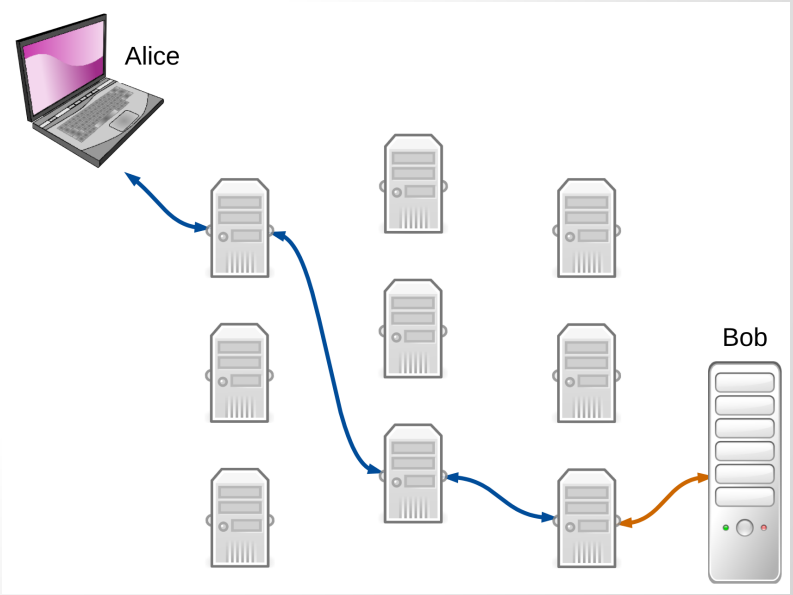
- Third-generation onion router
- Three-hop circuits
- Low latency
- Directory servers
  - Network status documents
- Very popular
  - 2.2 million daily users
  - 60 Gbits observed traffic



Background - Tor Routing

- Three routers: entry, middle, exit





Background - Hidden Services

- Servers hidden by the Tor network
- Websites of unknown location or ownership
- Bi-directional anonymity
- Only accessible through Tor
- Relatively popular
  - ~27,000 hidden services
  - 40 Mbits observed traffic

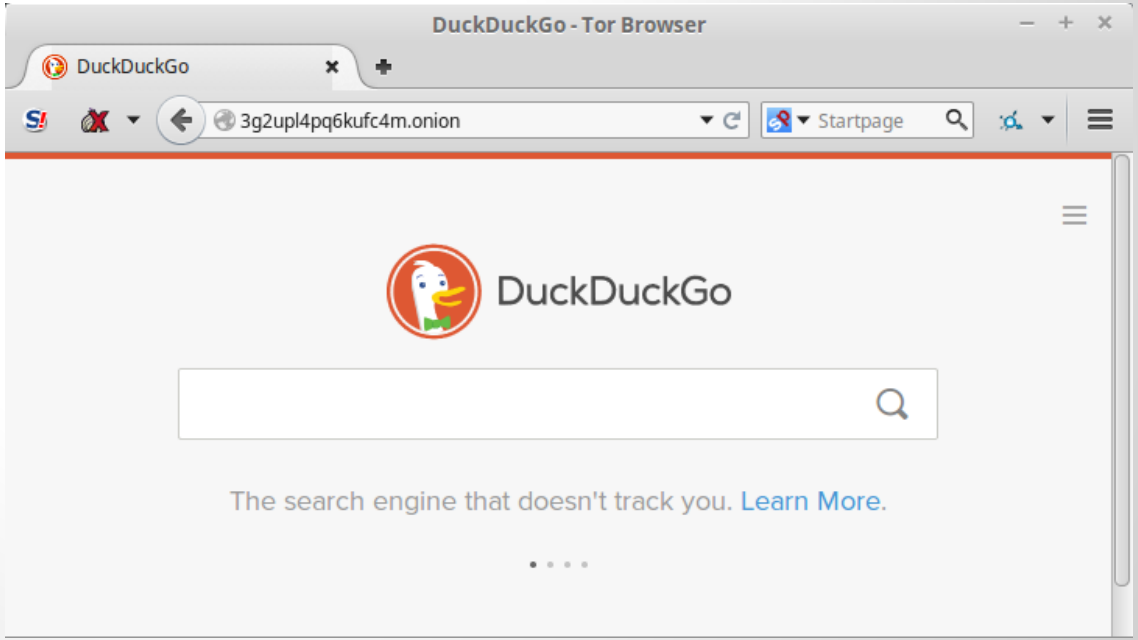
The Problem

- Hidden services have usability issues

3g2upl4pq6kufc4m.onion

33y6fjyhs3phzfjj.onion

vbmwh445kf3fs2v4.onion



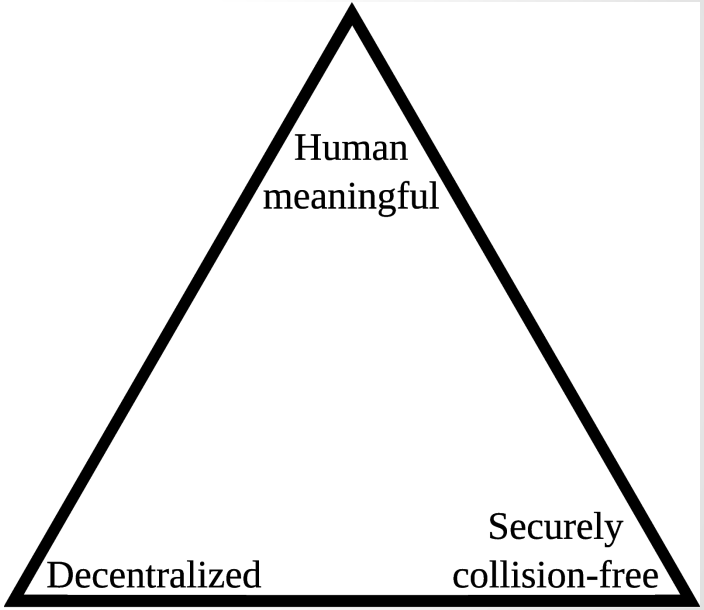
DNS Objectives

1. Anonymous registrations.
2. Privacy-enhanced lookups.
3. Authenticatable registrations.
4. Unique domain names.
5. Distributed design.
6. Simple and relatively easy to use.
7. Backwards compatibility.

These are not met by other existing works.

DNS Fundamental Challenges

- Zooko's Triangle
- Three properties
  - Can only have two
- Examples:
  - Human nicknames
  - Domain names
  - HS addresses



Contributions

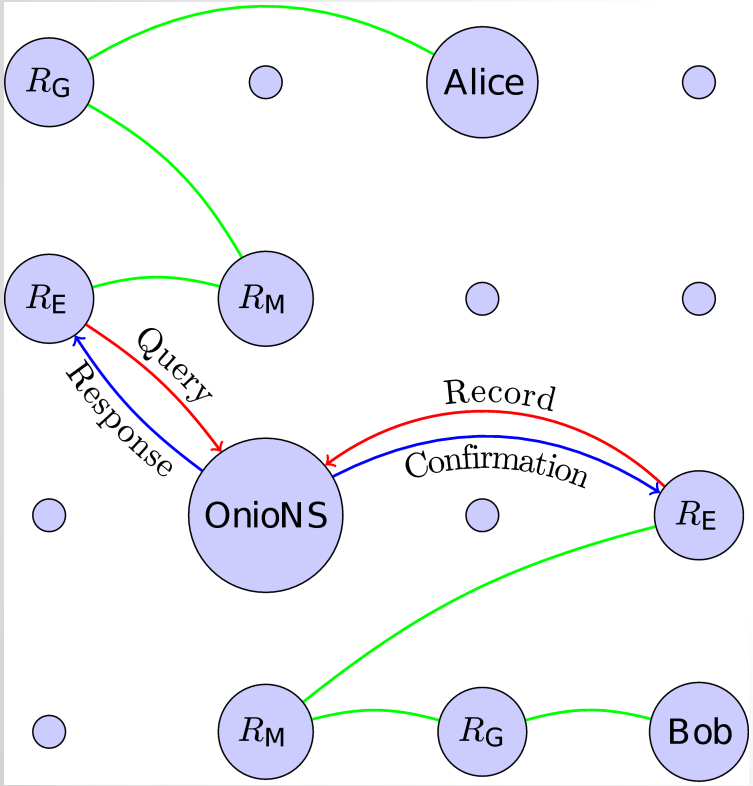
- Fixing of major usability issue within Tor
  - Allow hidden services to be accessed by meaningful domain names, rather than complicated addresses
- No central authority
  - Distributed design with new distributed self-healing database
- Privacy enhanced
  - Anonymous registration, anonymous lookup queries
- Easy integration into Tor’s infrastructure
  - Designed as a plugin, no significant changes to Tor
- Verifiable
  - Database and registrations can be authenticated

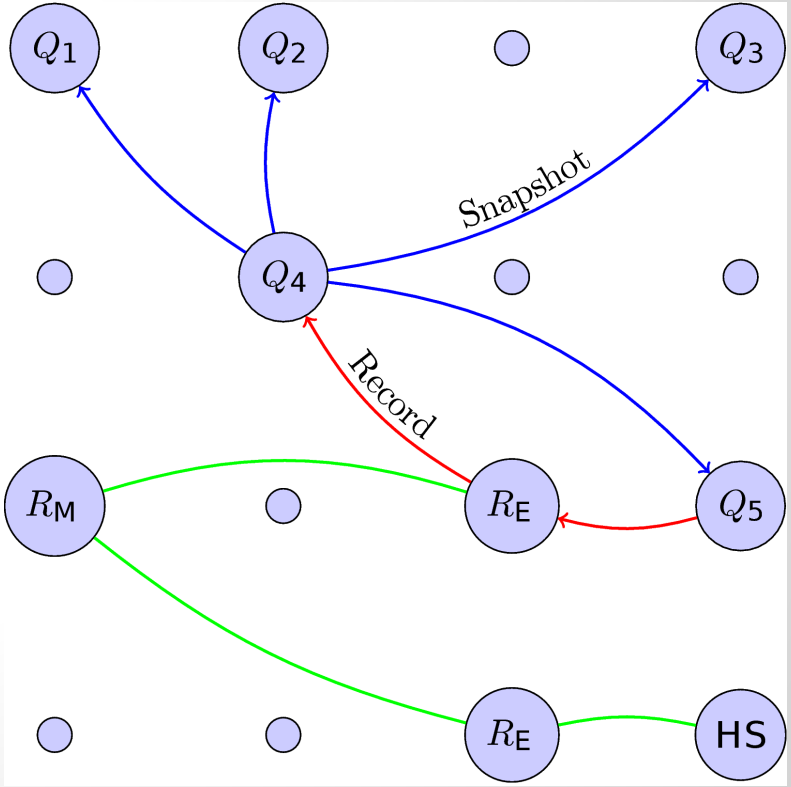
The Onion Name System (OnioNS)

- Distributed DNS inside the Tor network

1. Hidden service operator generates registration.
2. Operator sends registration to OnioNS.
3. Registration is flooded to all OnioNS participants.
4. Client performs query for domain name.
5. Client receives registration.
6. Client verifies registration and visits HS.

Design Overview





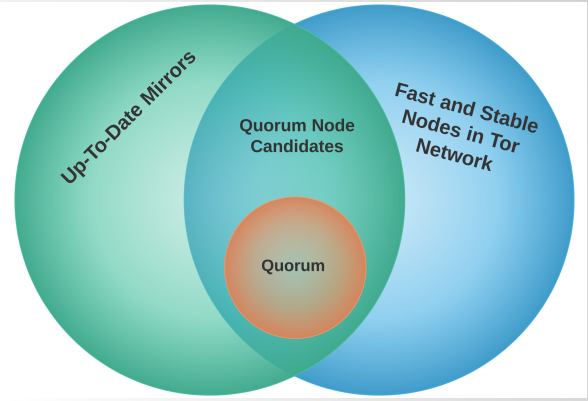
Data Structures - Record

- Holds .tor to .onion association
- Several types
  - Create, Modify, Move, Renew, Delete
- Self-signed by HS’s key
  - Signature can be matched against destination HS
- Some cost to generate
  - Proof-of-work - hard to find, easy to verify

Data Structures - Page

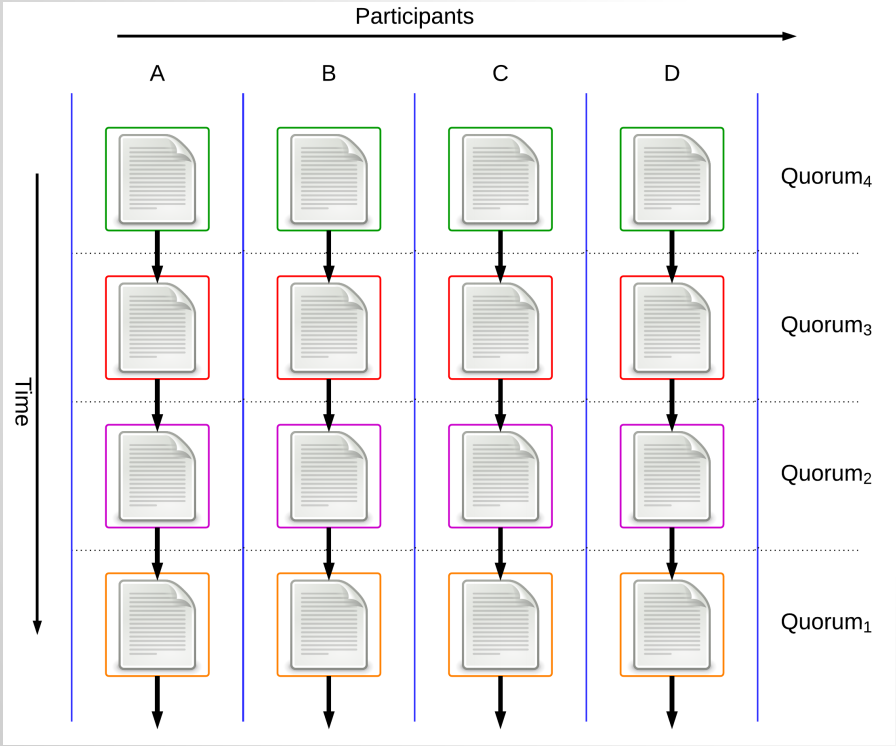
- Holds a list of Records
- Long-term data structure
- References a previous Page
  - Forming a Pagechain
- Read-only once final
- Similar in principle to BTC/NMC blockchain

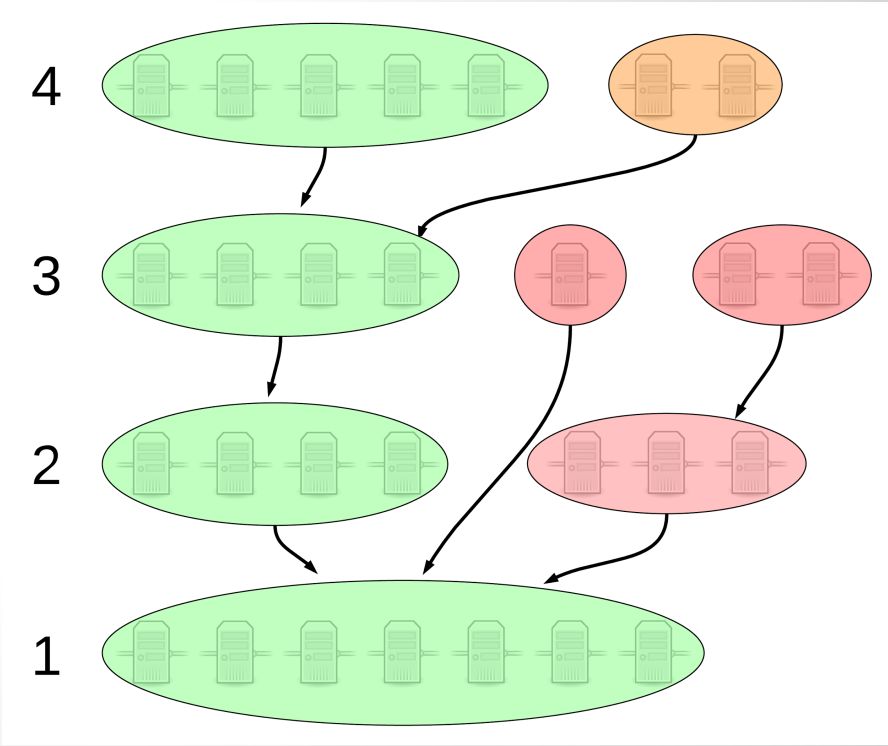
The Quorum



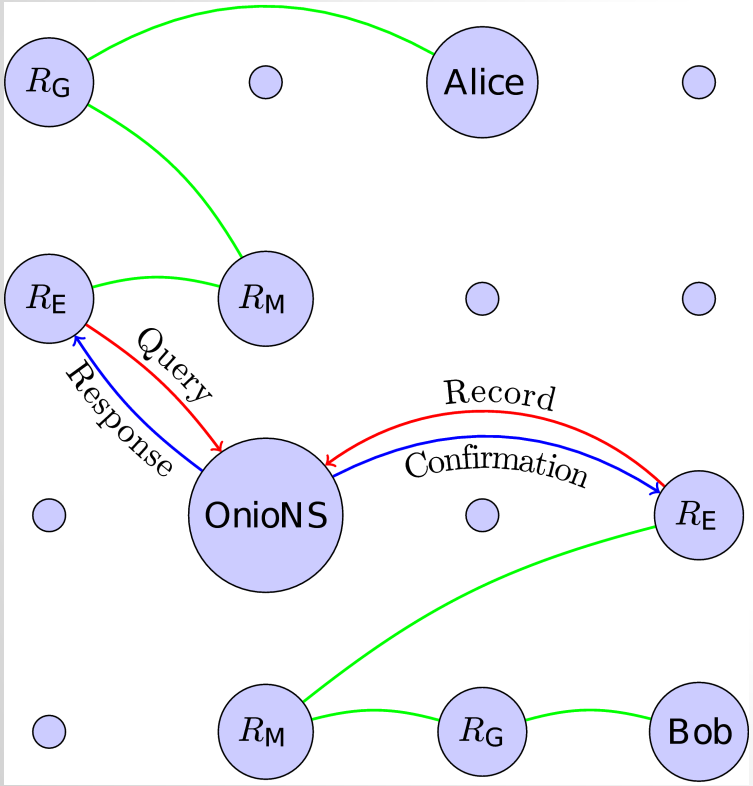
- New Records sent to Quorum
  - ~127 of them
  - Maintain head of Pagechain
  - Replaced periodically by new Quorum
- Participating Tor nodes hold Pagechain
  - Can verify Pagechain integrity & authenticity
  - Mirror off of Quorum nodes
  - Resolve client queries
  - Distributed, load-balanced

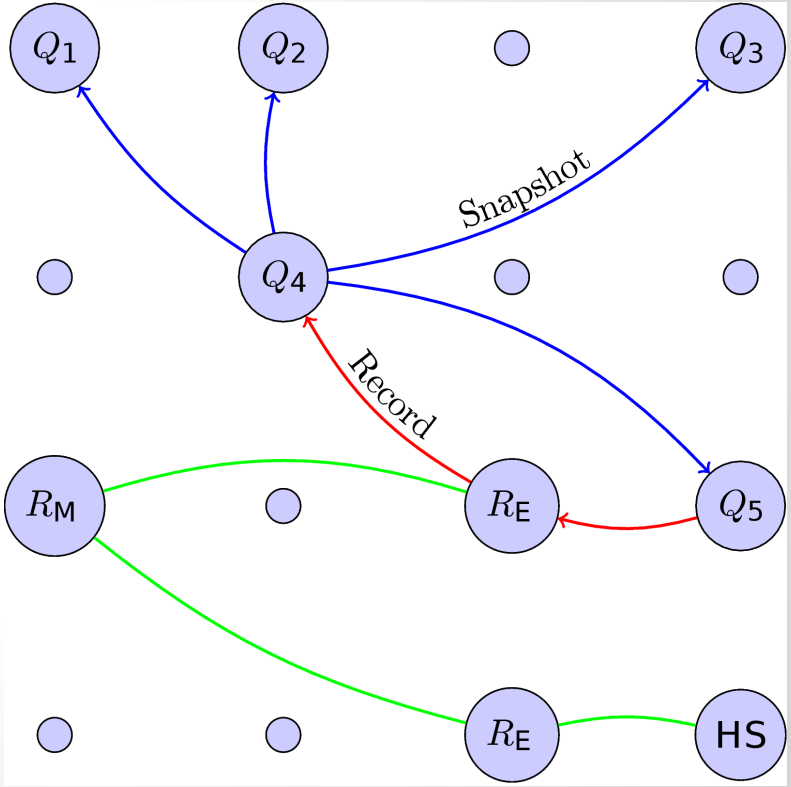
The Pagechain





Design Overview

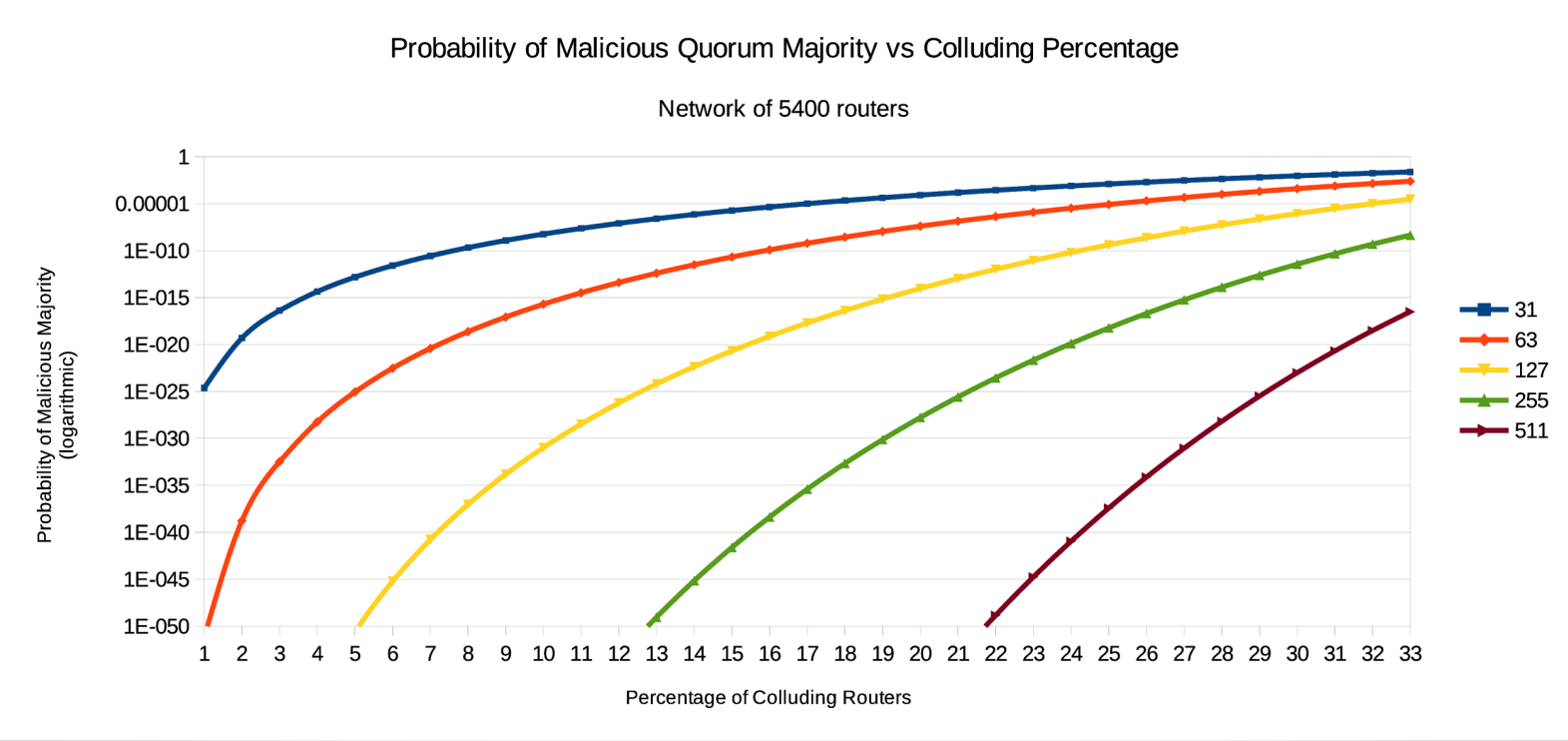




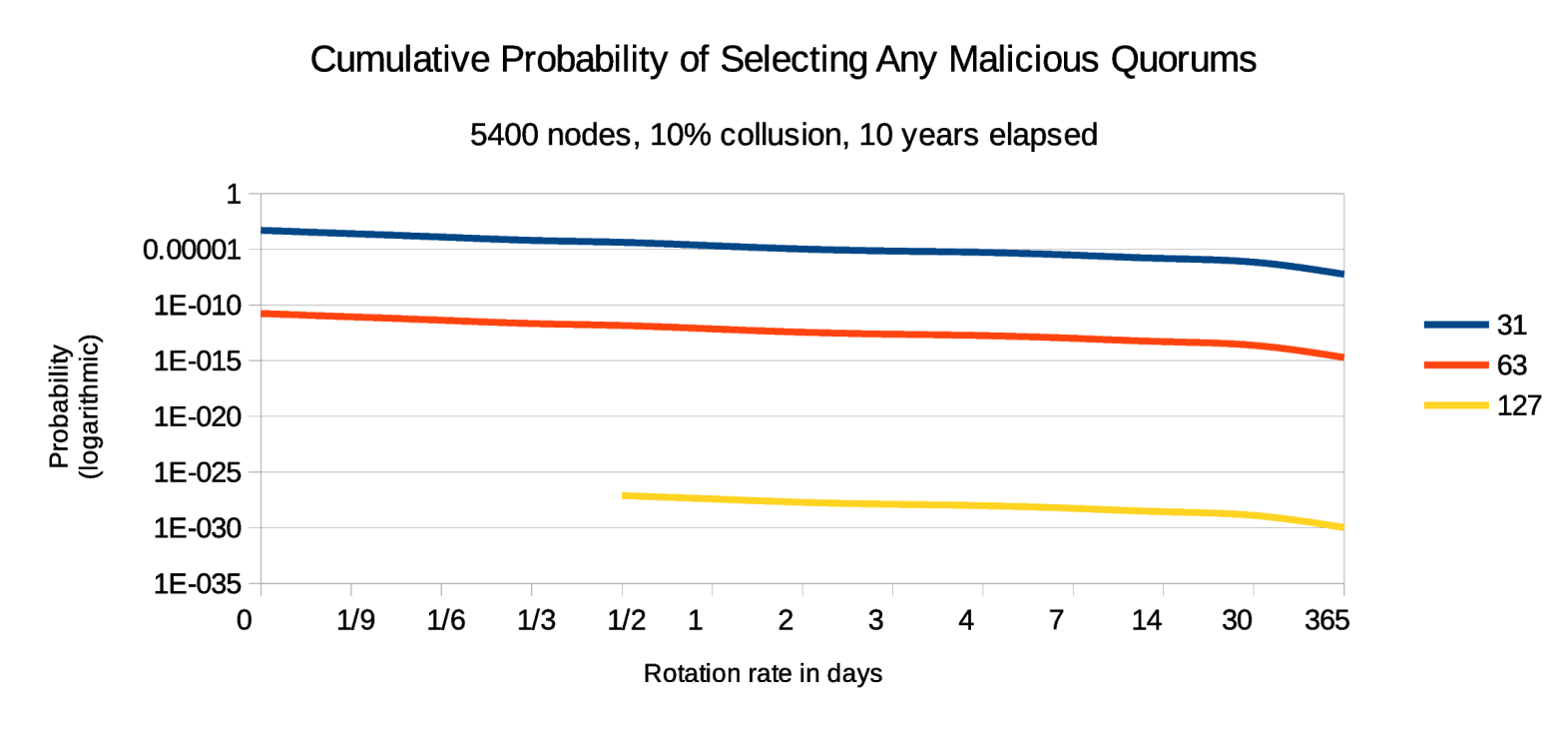
Security - Quorum Selection

- Assume that attacker controls some colluding Tor nodes
- Network follows largest agreeing subset
  - Attacker wins if > 50% of the Quorum
- Probability of compromise
  - Size of the Quorum
  - Selection frequency
- Can assume a statistical environment
  - Selection without replacement
  - Hypergeometric distribution

Security - Quorum Selection

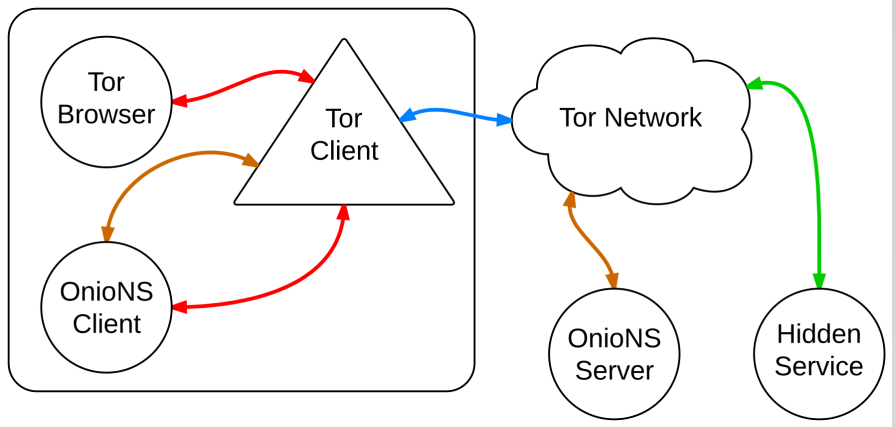


Security - Quorum Rotation



Prototype

- Simple client lookup, fixed resolver

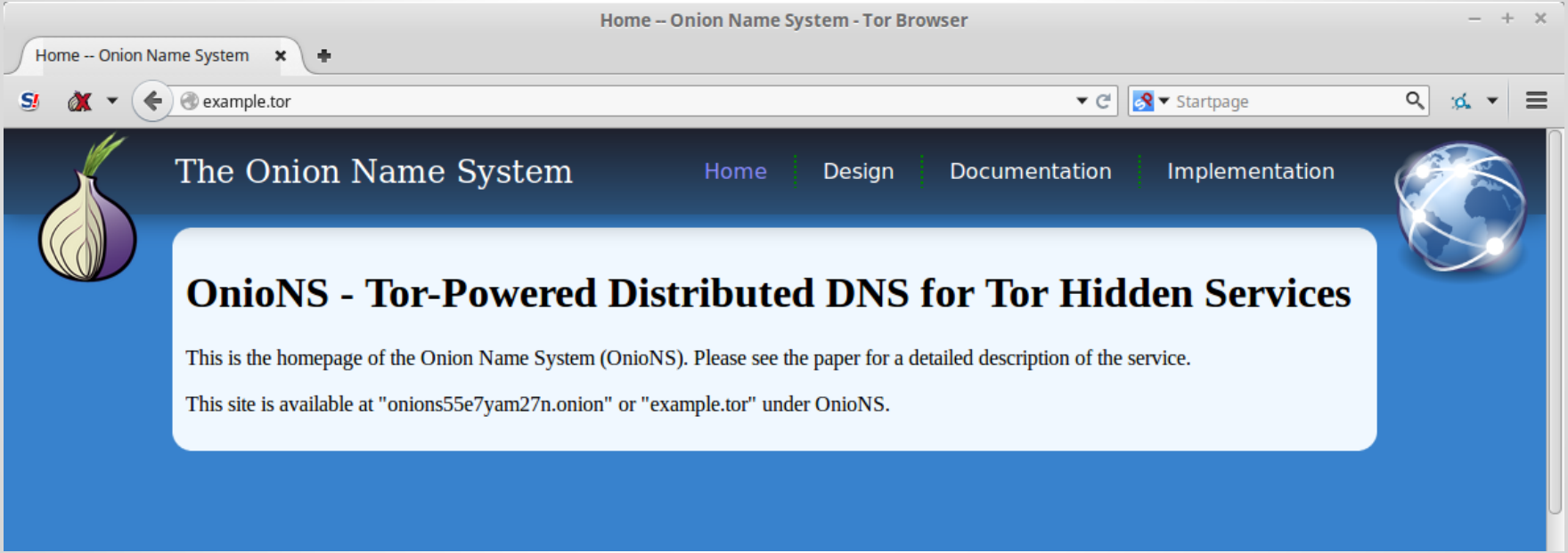


1. User types “*.tor” into Tor Browser
2. Tor client sends “*.tor” to OnioNS client
3. Client sends “*.tor” to resolver over Tor circuit
4. Resolver returns a Record
5. OnioNS client verifies Record, gives .onion to Tor
6. Tor performs lookup for the Tor Browser

Prototype - Demonstration

- Created a hidden service
  - onions55e7yam27n.onion - vanity key generator
- Create Record
  - “example.tor” -> “onions55e7yam27n.onion”
  - Resolver set to return this Record
- Completed
  - Tor modifications - Interception, Libevent
  - Tor-OnioNS IPC - Named pipes
  - Network programming - Boost Asio

Prototype - Demonstration



Prototype - Analysis

- Performance
  - Circuit dependent
  - 2-3 seconds (15 samples)
- Future work
  - Increase reliability
  - Dynamic resolvers
  - More client, HS, and server protocols
  - More realistic deployments
  - Tor’s SoP work

Conclusion

- The Onion Name System
- Addressing major HS usability issues
- Distributed design, no central authority
- Verifiable data structures
- Secure under design assumptions
- Privacy enhanced
  - No identity leakage of HS or client
- Easy integration into Tor’s infrastructure