Anonymity With Tor

The Onion Router

Nathan S. Evans   Christian Grothoff

Technische Universität München

July 5, 2012

"It's a series of tubes." – Ted Stevens

Overview

- What is Tor?
- Motivation
- Background Material
- How Tor Works
- Hidden Services
- Attacks
- Specific Attack
- Summary

What is Tor?

- Tor is a *P2P network* of Chaum inspired *low-latency mixes* which are used to provide *anonymous* communication between parties on the Internet.

What is Tor?

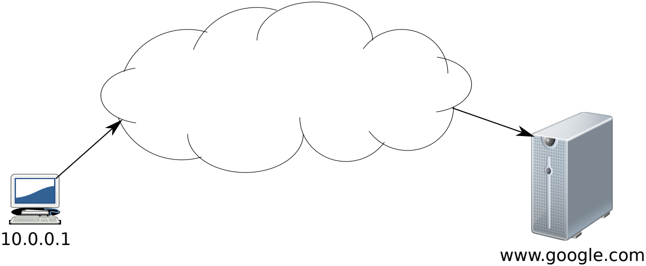
- Sender anonymity for low latency applications
- Common usage: Web browsing
  - Sender anonymity
  - Web server cannot identify client
- Advanced usage:
  - Hidden services (send/receive anonymity)
  - Filesharing
  - IRC
  - Any application that communicates using TCP

⇒ Tor provides users with a service that effectively hides their identity on the Internet.

Motivation

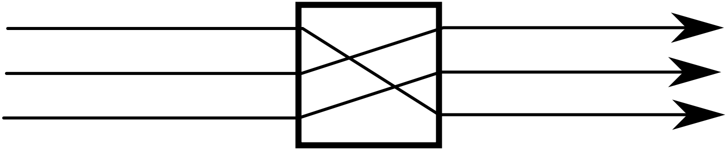
- Internet packets travel from A to B transparently
- A knows B, and B knows A (by IP address)
- Routers, etc. can determine that A and B are communicating
- This may reveal unintended information (e.g. person X's bank)
- Encryption
  - For example, TLS (HTTPS)
  - Provides *Data anonymity*
  - Does not hide routing information

Motivation - Routing Example



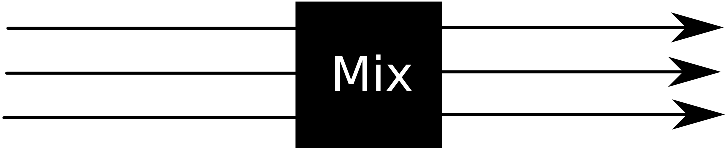
Review: Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



Review: Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:

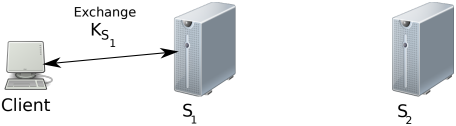


Onion Routing

- Multiple mix servers
- Subset of mix servers chosen by initiator
- Chosen mix servers create “circuit”
  - Initiator contacts first server S_1, sets up symmetric key K_{S_1}
  - Then asks first server to connect to second server S_2; through this connection sets up symmetric key with second server K_{S_2}
  - ...
  - Repeat with server S_i until circuit of desired length n constructed

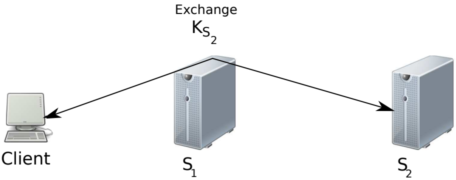
Onion Routing Example

- Client sets up symmetric key K_{S_1} with server S_1



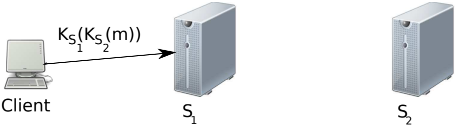
Onion Routing Example

- Via S_1 Client sets up symmetric key K_{S_2} with server S_2



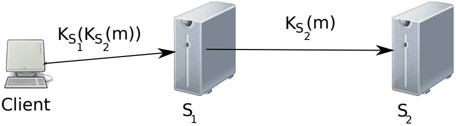
Onion Routing Example

- Client encrypts m as $K_{S_1}(K_{S_2}(m))$ and sends to S_1



Onion Routing Example

- S_1 decrypts, sends on to S_2, S_2 decrypts, revealing m

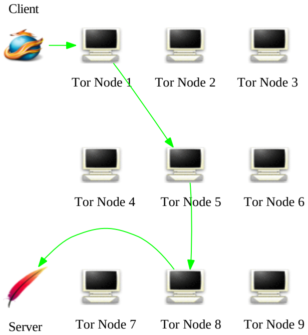


Tor - How it Works

- Low latency P2P Network of mix servers
- Designed for interactive traffic (https, ssh, etc.)
- "Directory Servers" store list of participating servers
  - Contact information, public keys, statistics
  - Directory servers are replicated for security
- Clients choose servers randomly with bias towards high BW/uptime
- Clients build long lived Onion routes "circuits" using these servers
- Circuits are bi-directional
- Circuits are hard coded at length three

Tor - How it Works - Example

- Example of Tor client circuit



Tor - How it Works - Servers

- Servers connected in "full mesh"
  - All servers exchange symmetric keys
  - Allows fast sending between servers, regardless of which circuits
  - Allows combining of multiple messages with same next-hop
- New servers publish information to directory servers
- Once online for a certain period, they are added to the "live" list
- They are then available for use by clients

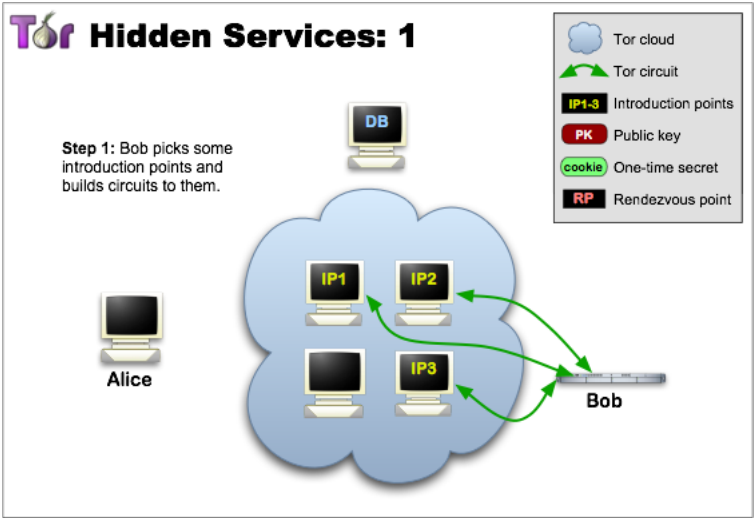
Tor - How it Works - Servers

- Servers are classified into three categories for usability, security and operator preference
- Entry nodes (aka guards) - chosen for first hop in circuit
    - Generally long lived "good" nodes
    - Small set chosen by client which are used for client lifetime (security)
- Middle nodes - chosen for second hop in circuit, least restricted set
- Exit nodes - last hop in circuit
    - Visible to outside destination
    - Support filtering of outgoing traffic
    - Most vulerable position of nodes

Hidden Services in Tor

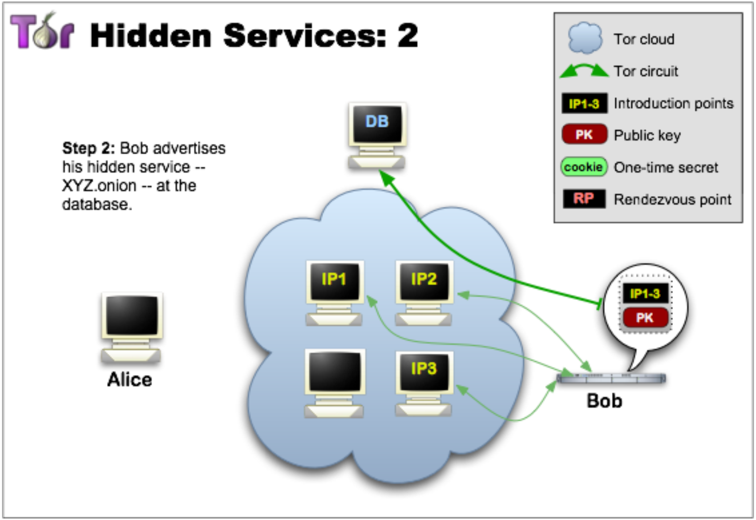
- Hidden services allow Tor servers to receive incoming connections anonymously
- Can provide access to services available *only* via Tor
  - Web, IRC, etc.
  - For example, host a website without your ISP knowing
- Uses a "Rendezvous point“ to connect two Tor circuits
- Uses "Introduction points“, which allow outside peers to contact hidden server (while keeping it hidden)
- Publishes Intro. point addresses to "Lookup server“
- Client gets Introduction point address from lookup server, sends random rendezvous point to hidden server
- Data travels a total of 7 hops (once established)

Hidden Services Example 1



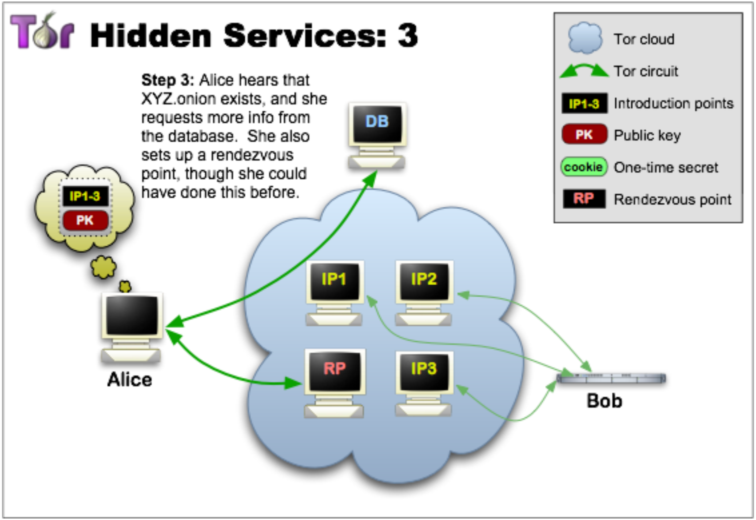
Step 1: Bob picks some introduction points and builds circuits to them.

Hidden Services Example 2



Step 2: Bob advertises his hidden service -- XYZ.onion -- at the database.

Hidden Services Example 3

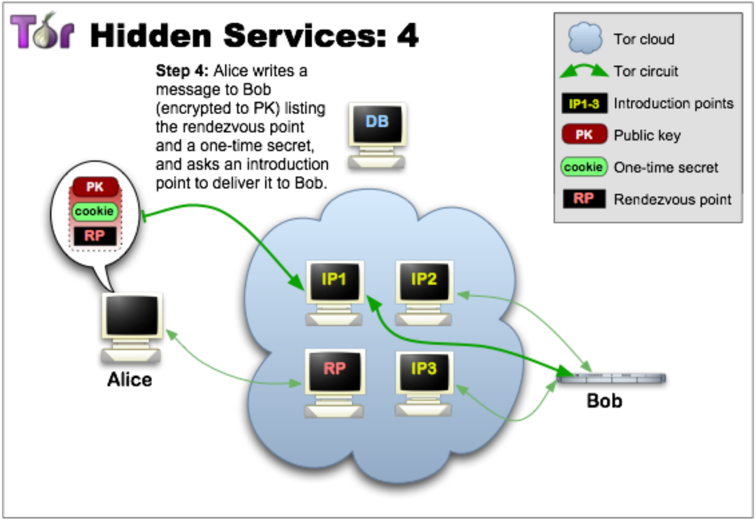


Hidden Services: 3

Step 3: Alice hears that XYZ.onion exists, and she requests more info from the database. She also sets up a rendezvous point, though she could have done this before.

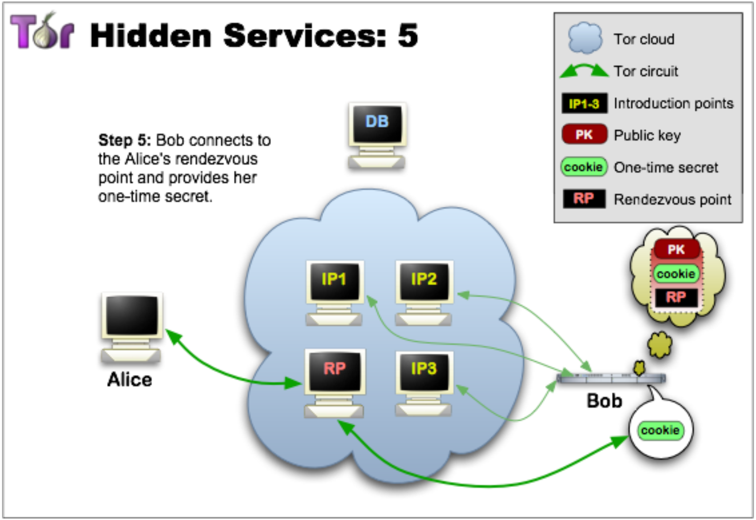
☁	Tor cloud
⤺	Tor circuit
IP1-3	Introduction points
PK	Public key
cookie	One-time secret
RP	Rendezvous point

Hidden Services Example 4



Step 4: Alice writes a message to Bob (encrypted to PK) listing the rendezvous point and a one-time secret, and asks an introduction point to deliver it to Bob.

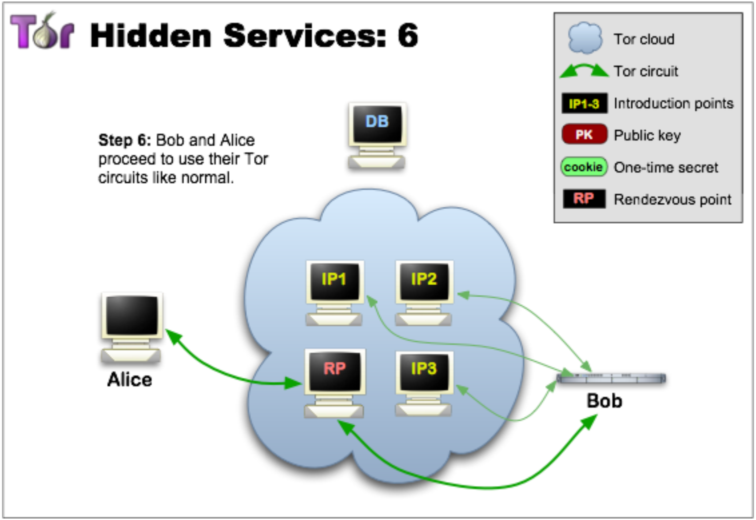
Hidden Services Example 5



Hidden Services: 5

Step 5: Bob connects to the Alice's rendezvous point and provides her one-time secret.

Hidden Services Example 6



Hidden Services: 6

Step 6: Bob and Alice proceed to use their Tor circuits like normal.

Types of Attacks on Tor

- Exit Relay Snooping
- Website fingerprinting
- Traffic Analysis
- Intersection Attack
- DoS

Why attack Tor?

- Tor is the most popular and widely used free software P2P network used to achieve anonymity on the Internet:
    - Tor has a large user base
    - The project is well supported
    - Generally assumed to give users strong anonymity

Our results:

All the Tor nodes involved in a circuit can be discovered, reducing Tor users level of anonymity and revealing a problem with Tor's protocol

Key Tor Properties

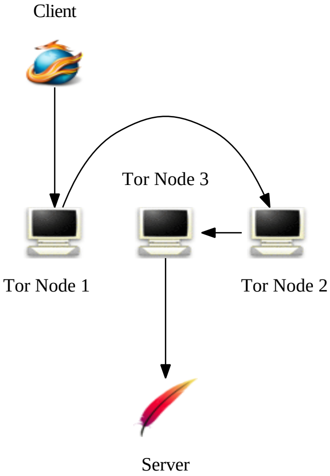
- Data is forwarded through the network
- Each node knows only the previous hop and the next hop
- Only the originator knows all the hops
- Number of hops is hard coded (currently set to three)

Key security goal: No node in the path can discover the full path

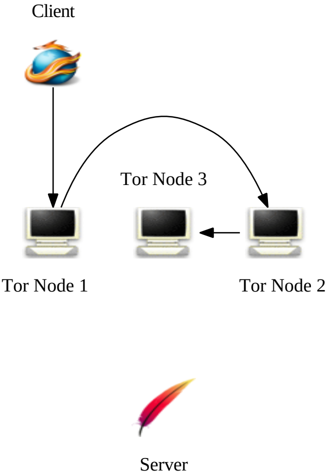
Our Basis for Deanonymization

- Target user is running Tor from 2009 with default settings
- Three design issues enable users to be deanonymized
  1. No artificial delays induced on connections
  2. Path length is set at a small finite number (3)
  3. Paths of arbitrary length through the network can be constructed

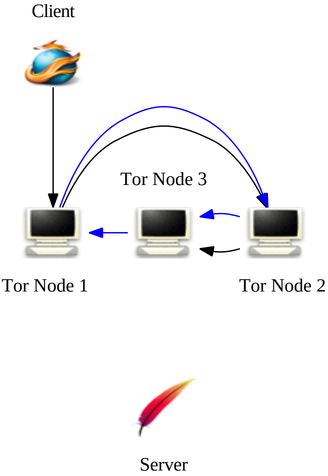
Regular Path Example



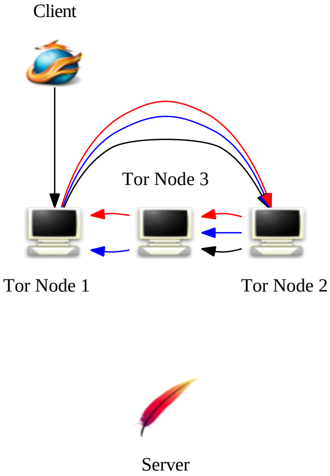
Circular Path Example 1/5



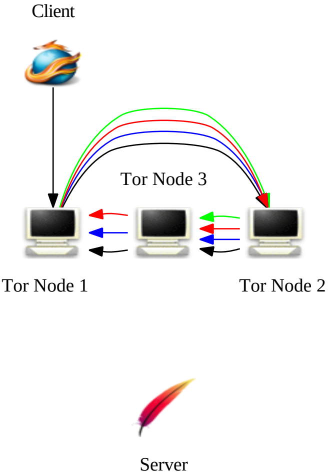
Circular Path Example 2/5



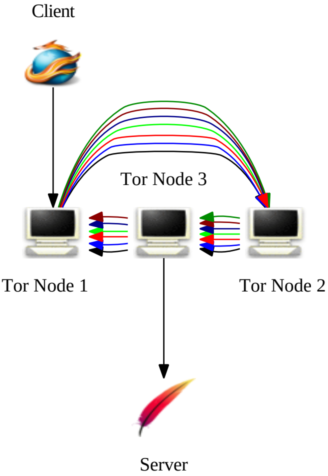
Circular Path Example 3/5



Circular Path Example 4/5



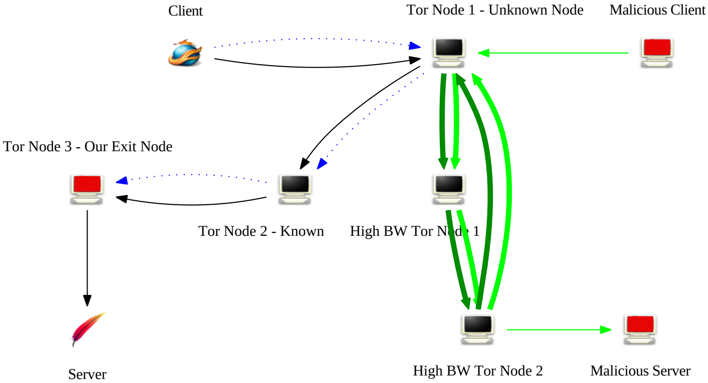
Circular Path Example 5/5



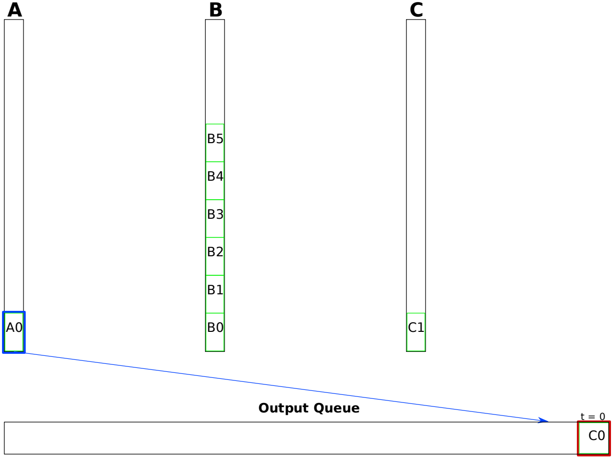
Attack Implementation

- Exit node “injects” JavaScript “ping” code into HTML response
- Client browses as usual, while JavaScript continues to “phone home”
- Exit node measures variance in latency
- While continuing to measure, attack strains possible first hop(s)
- If no significant variance observed, pick another node from candidates and start over
- Once sufficient change is observed in *repeated* measurements, initial node has been found

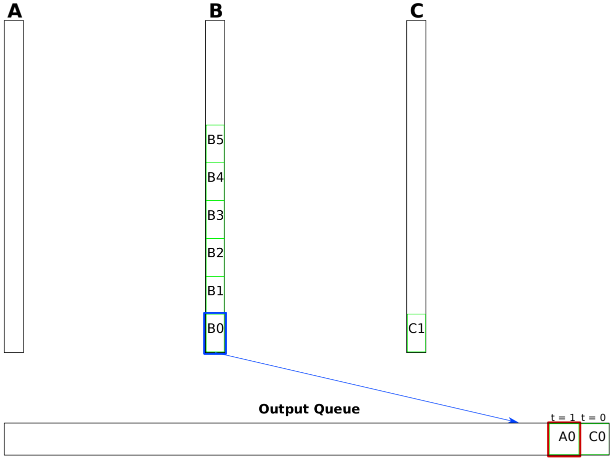
Attack Example



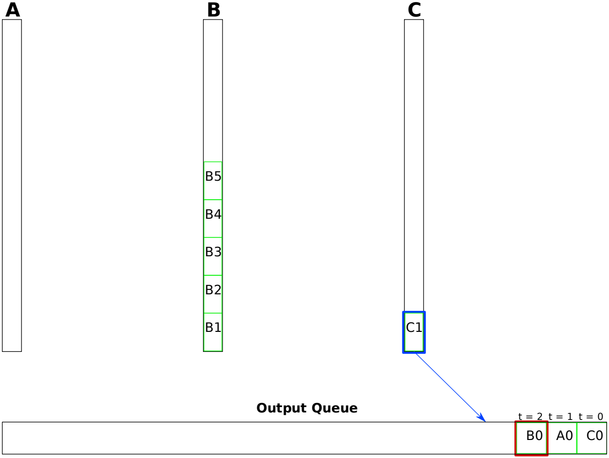
Queue example 1 (3 circuits)



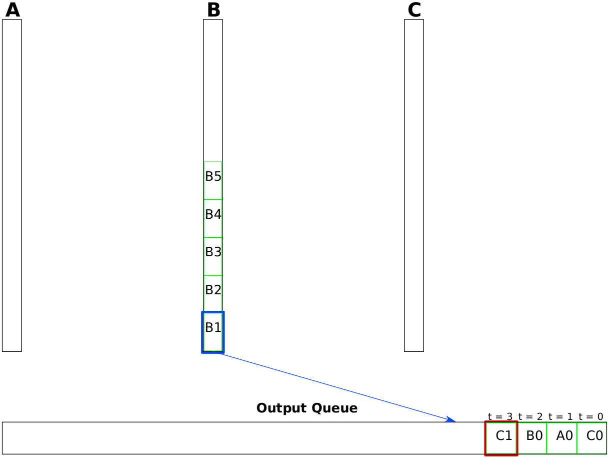
Queue example 2 (3 circuits)



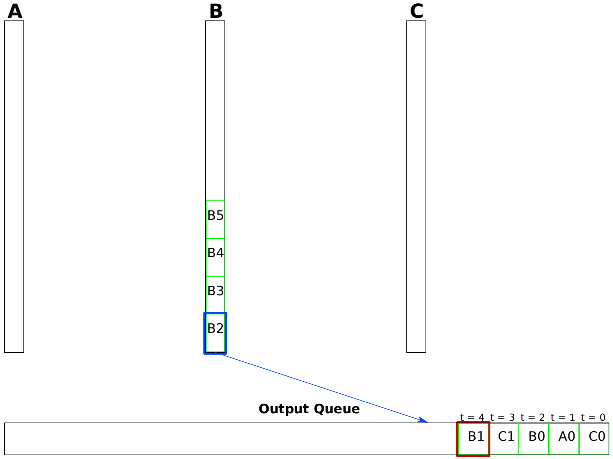
Queue example 3 (3 circuits)



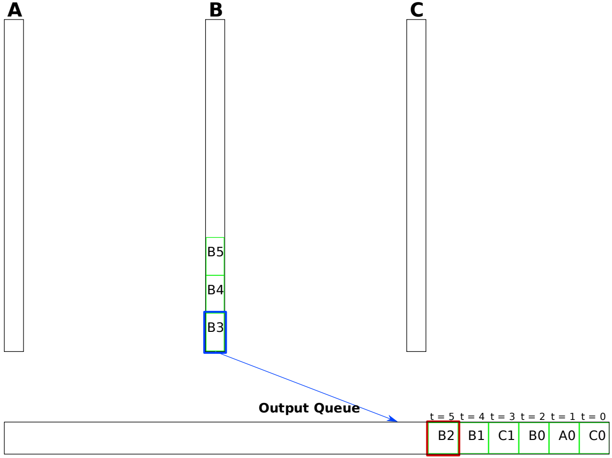
Queue example 4 (3 circuits)



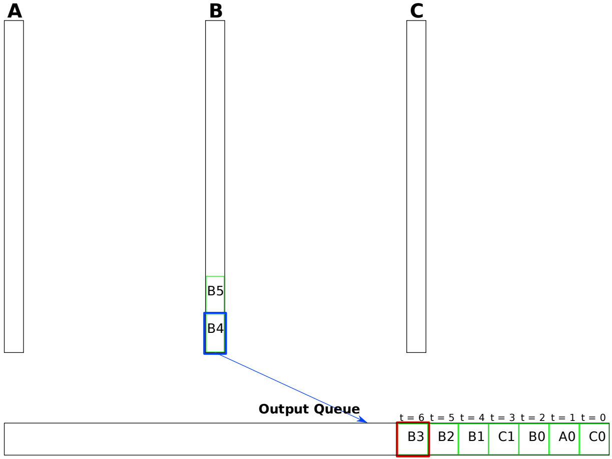
Queue example 5 (3 circuits)



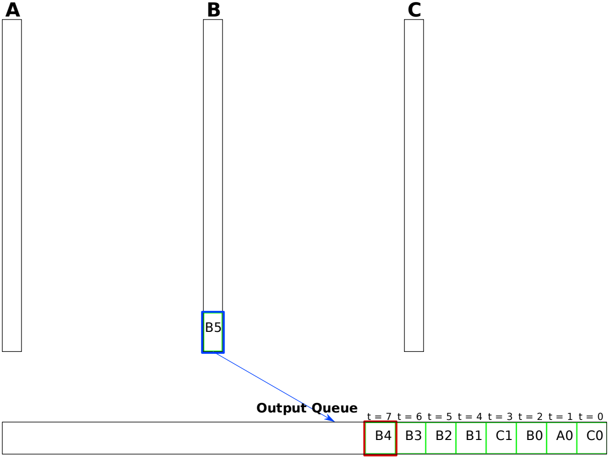
Queue example 6 (3 circuits)



Queue example 7 (3 circuits)

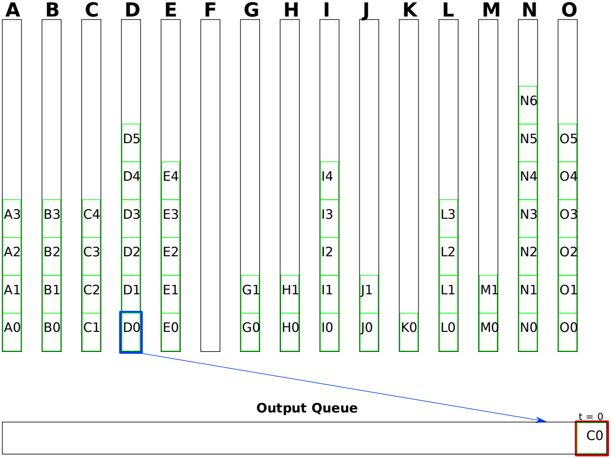


Queue example 8 (3 circuits)

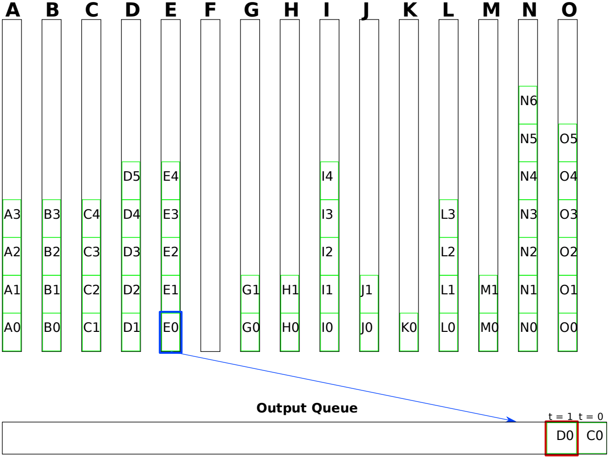


t = 7	t = 6	t = 5	t = 4	t = 3	t = 2	t = 1	t = 0
B4	B3	B2	B1	C1	B0	A0	C0

Queue example 1 (15 circuits)



Queue example 2 (15 circuits)

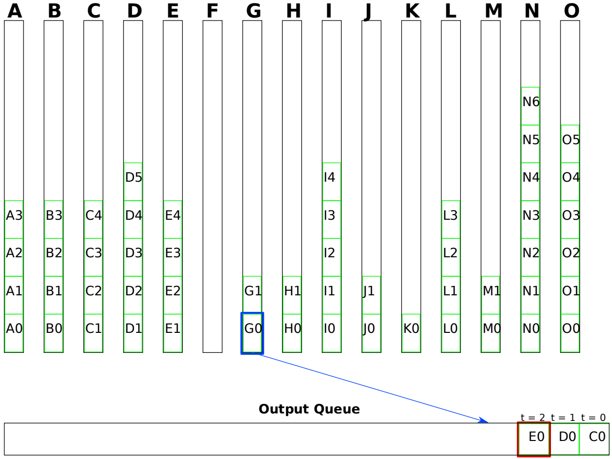


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
													N6	
													N5	O5
			D5	E4				I4					N4	O4
A3	B3	C4	D4	E3				I3			L3		N3	O3
A2	B2	C3	D3	E2				I2			L2		N2	O2
A1	B1	C2	D2	E1		G1	H1	I1	J1		L1	M1	N1	O1
A0	B0	C1	D1	E0		G0	H0	I0	J0	K0	L0	M0	N0	O0

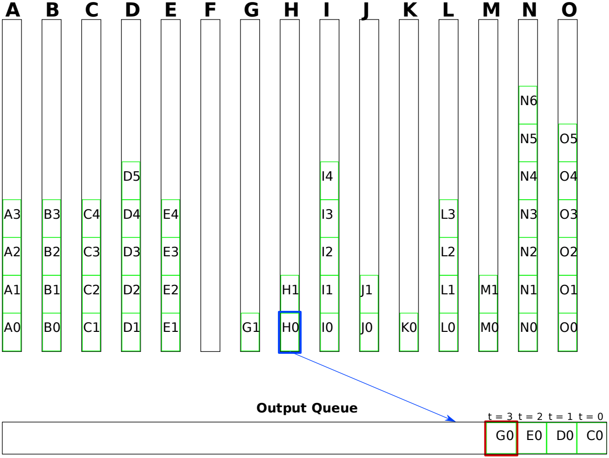
Output Queue

t = 1	t = 0
D0	C0

Queue example 3 (15 circuits)



Queue example 4 (15 circuits)

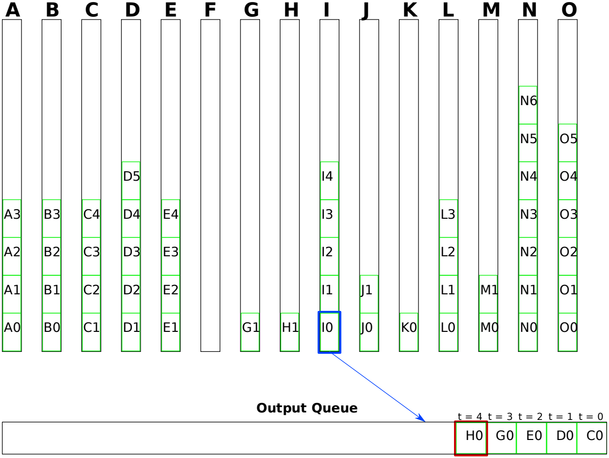


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
													N6	
													N5	O5
			D5					I4					N4	O4
A3	B3	C4	D4	E4				I3			L3		N3	O3
A2	B2	C3	D3	E3				I2			L2		N2	O2
A1	B1	C2	D2	E2			H1	I1	J1		L1	M1	N1	O1
A0	B0	C1	D1	E1		G1	H0	I0	J0	K0	L0	M0	N0	O0

Output Queue

t = 3	t = 2	t = 1	t = 0
G0	E0	D0	C0

Queue example 5 (15 circuits)

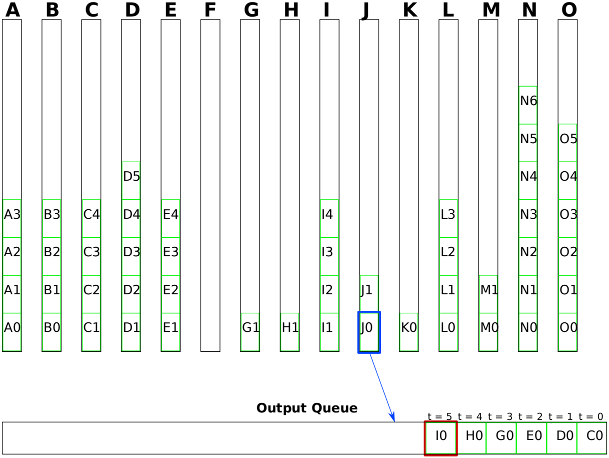


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
													N6	
													N5	O5
			D5					I4					N4	O4
A3	B3	C4	D4	E4				I3			L3		N3	O3
A2	B2	C3	D3	E3				I2			L2		N2	O2
A1	B1	C2	D2	E2				I1	J1		L1	M1	N1	O1
A0	B0	C1	D1	E1		G1	H1	I0	J0	K0	L0	M0	N0	O0

Output Queue

t = 4	t = 3	t = 2	t = 1	t = 0
H0	G0	E0	D0	C0

Queue example 6 (15 circuits)

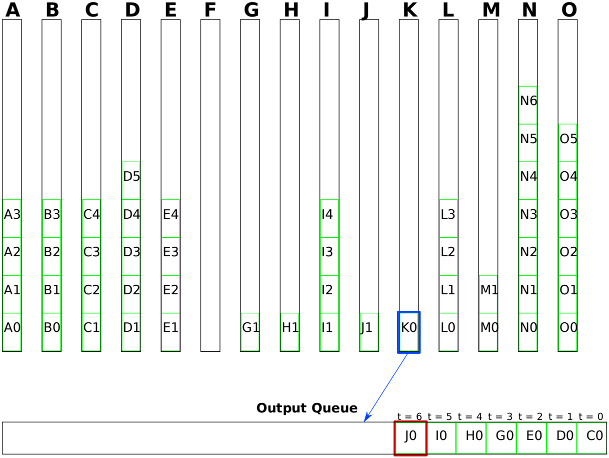


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
													N6	
													N5	O5
			D5										N4	O4
A3	B3	C4	D4	E4				I4			L3		N3	O3
A2	B2	C3	D3	E3				I3			L2		N2	O2
A1	B1	C2	D2	E2				I2	J1		L1	M1	N1	O1
A0	B0	C1	D1	E1		G1	H1	I1	J0	K0	L0	M0	N0	O0

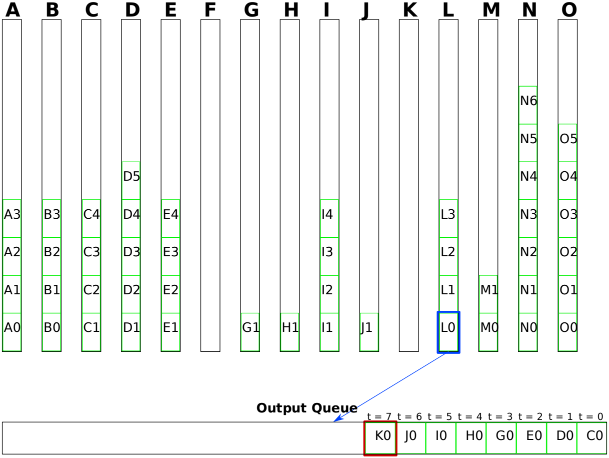
Output Queue

t = 5	t = 4	t = 3	t = 2	t = 1	t = 0
I0	H0	G0	E0	D0	C0

Queue example 7 (15 circuits)



Queue example 8 (15 circuits)

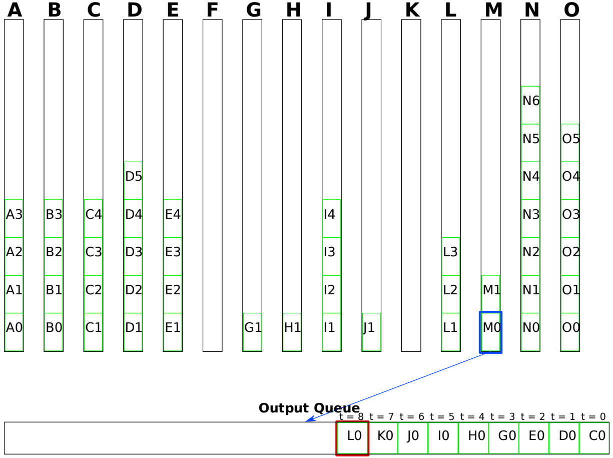


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
													N6	
													N5	O5
			D5										N4	O4
A3	B3	C4	D4	E4				I4			L3		N3	O3
A2	B2	C3	D3	E3				I3			L2		N2	O2
A1	B1	C2	D2	E2				I2			L1	M1	N1	O1
A0	B0	C1	D1	E1		G1	H1	I1	J1		L0	M0	N0	O0

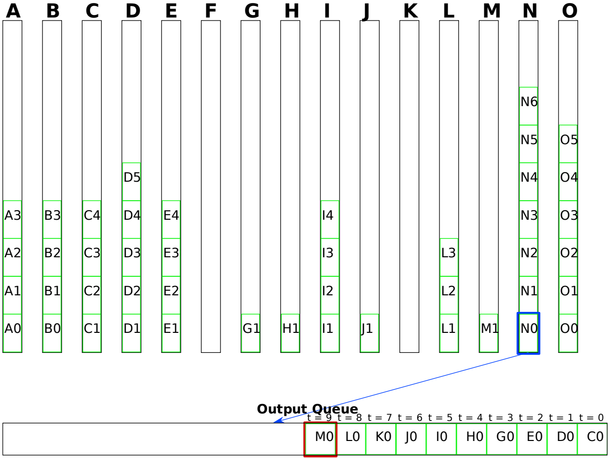
Output Queue

t = 7	t = 6	t = 5	t = 4	t = 3	t = 2	t = 1	t = 0
K0	J0	I0	H0	G0	E0	D0	C0

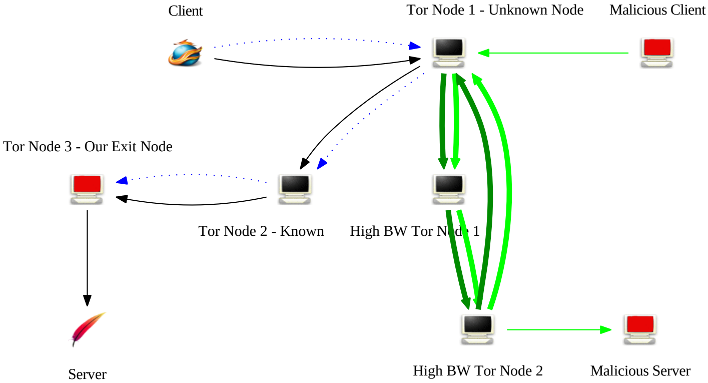
Queue example 9 (15 circuits)



Queue example 10 (15 circuits)



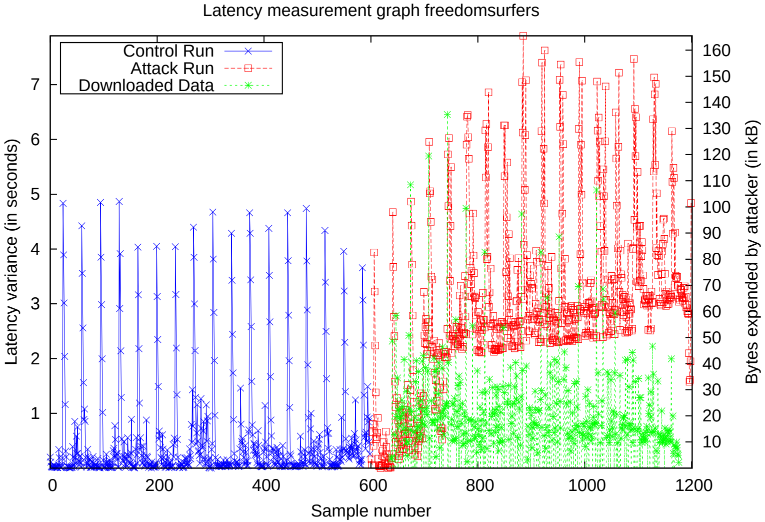
Attack Example



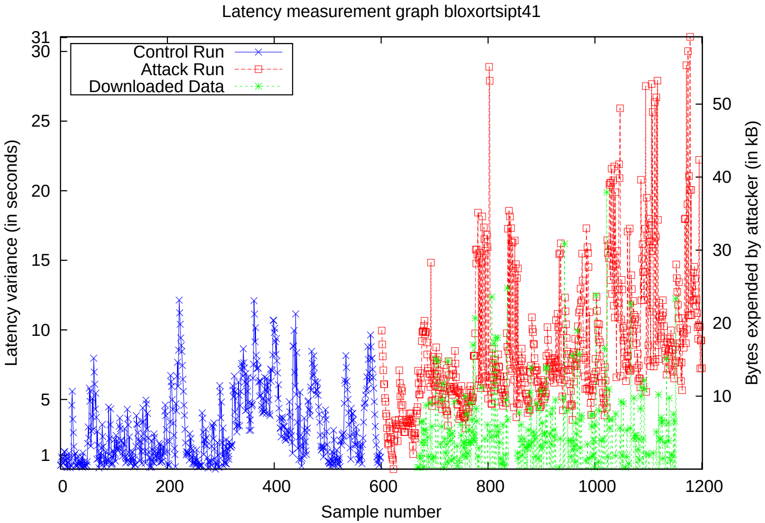
Attack Implementation

- Modified exit node
- Modified malicious client node
- Lightweight malicious web server running on GNU libmicrohttpd
- Client side JavaScript for latency measurements
- Instrumentation client to receive data

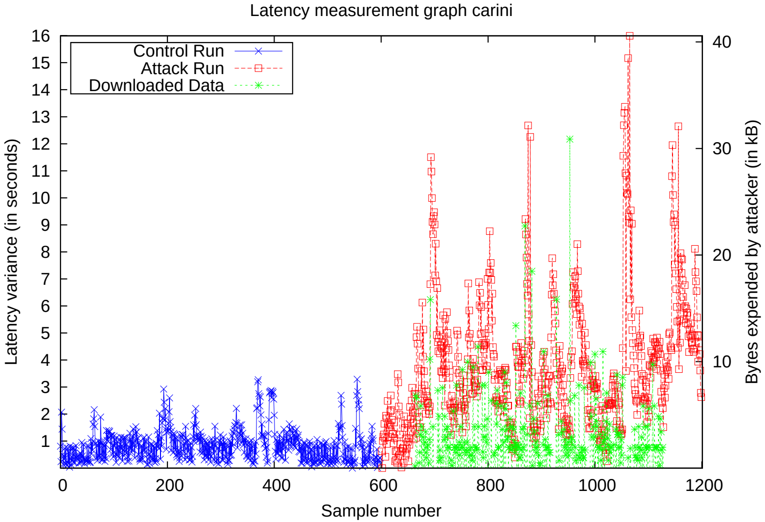
Gathered Data Example (1/8)



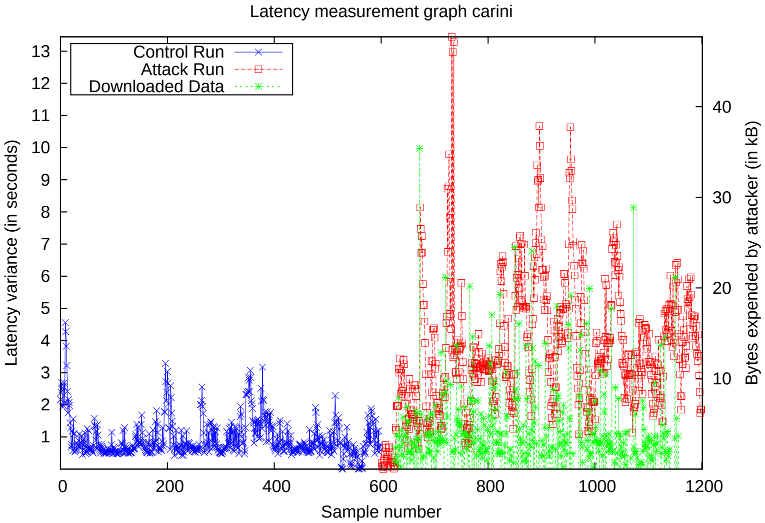
Gathered Data Example (2/8)



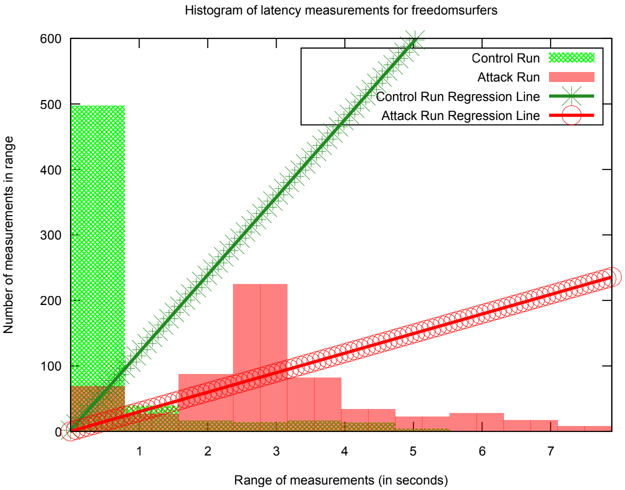
Gathered Data Example (3/8)



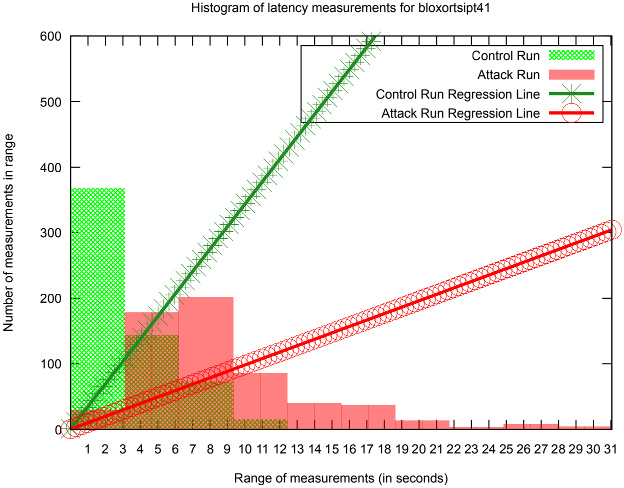
Gathered Data Example (4/8)



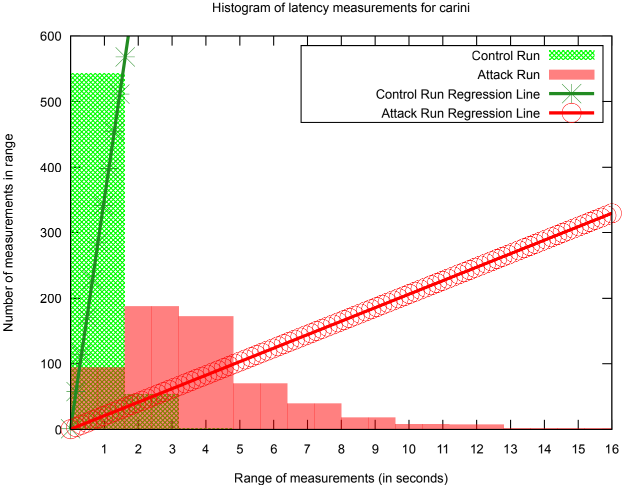
Gathered Data Example (5/8)



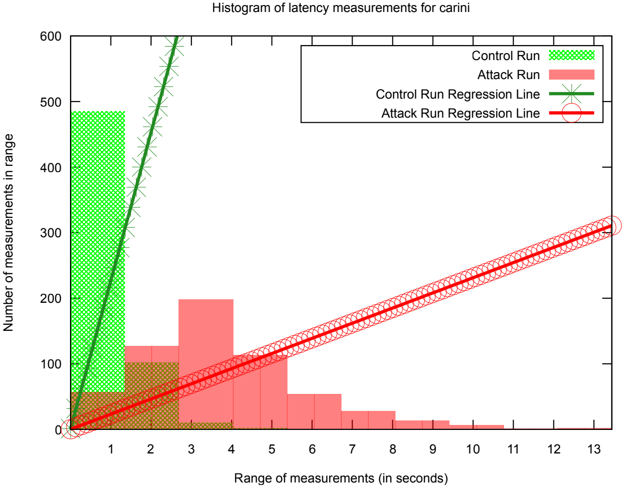
Gathered Data Example (6/8)



Gathered Data Example (7/8)



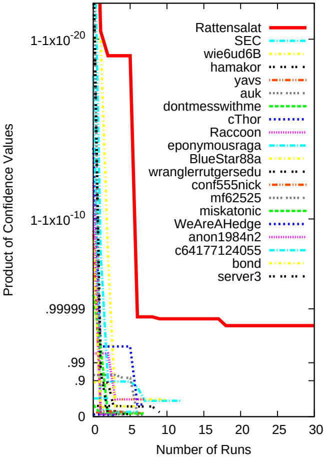
Gathered Data Example (8/8)

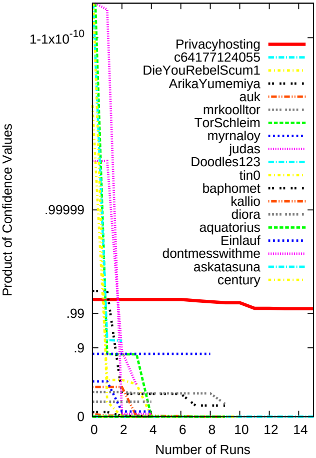


Statistical Analysis

- Use modified χ^2 test
- Compare baseline distribution to attack distribution
- High χ^2 value indicates distribution changed *in the right direction*
- Product of χ^2 confidence values over multiple runs
- Iterate over suspect routers until single node stands out

Cumulative Product of χ^2 p-values





What We Actually Achieve

- We do identify the entire path through the Tor network
- We do achieve this on the 2009 Tor network
- Attack works on routers with differing bandwidths
- This means that if someone were performing this attack from an exit node, Tor becomes as effective as a network of one-hop proxies

Why Our Attack is Effective

- Since we run the exit router, only a single node needs to be found
- Our multiplication of bandwidth technique allows low bandwidth connections to DoS high bandwidth connections (solves common DoS limitation)

Fixes

- Don't use a fixed path length (or at least make it longer)
- Don't allow infinite path lengths (this is fixed in Tor now!)
- Induce delays into connections (probably not going to happen)
- Monitor exit nodes for strange behavior (been done somewhat)
- Disable JavaScript in clients
- Use end-to-end encryption

Attack Improvements/Variants

- Use meta refresh tags for measurements instead of JavaScript
- Parallelize testing (rule out multiple possible first nodes at once)
- Improved latency measures for first hop to further narrow possible first hops

Conclusion

- Initial Tor implementation allowed arbitrary length paths
- Arbitrary path lengths allow latency altering attack
- Latency altering attack allows detection of significant changes in latency
- Significant changes in latency reveal paths used

Questions?

?