

Trust-based Anonymous Communication: Models and Routing Algorithms

Aaron Johnson *U.S. Naval Research Laboratory*

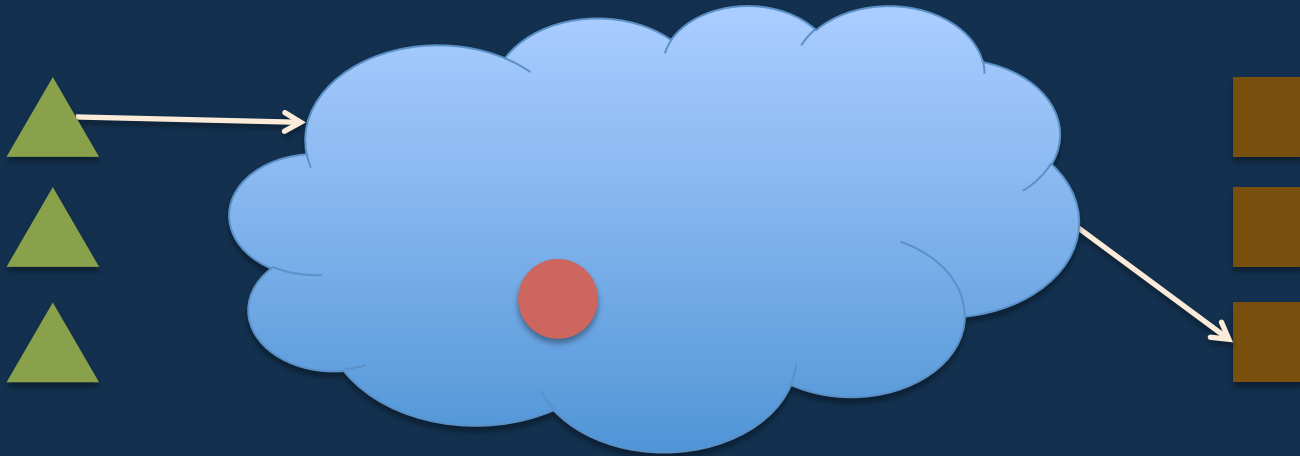
Paul Syverson *U.S. Naval Research Laboratory*

Roger Dingledine *The Tor Project*

Nick Mathewson *The Tor Project*

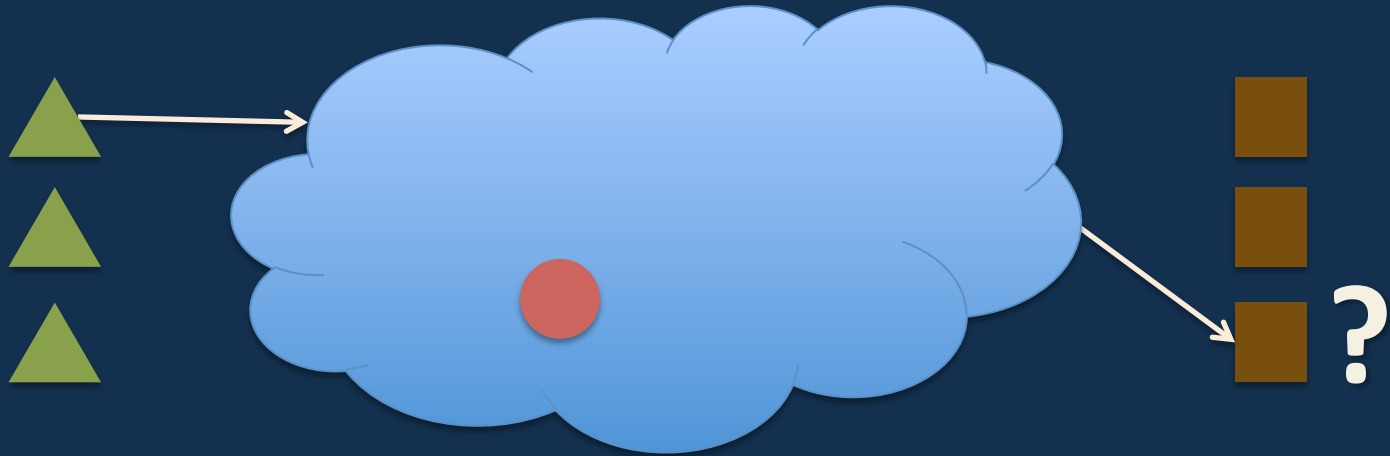
18th ACM Conference on Computer and Communications Security
October 17-21, 2011
Chicago, IL

Overview



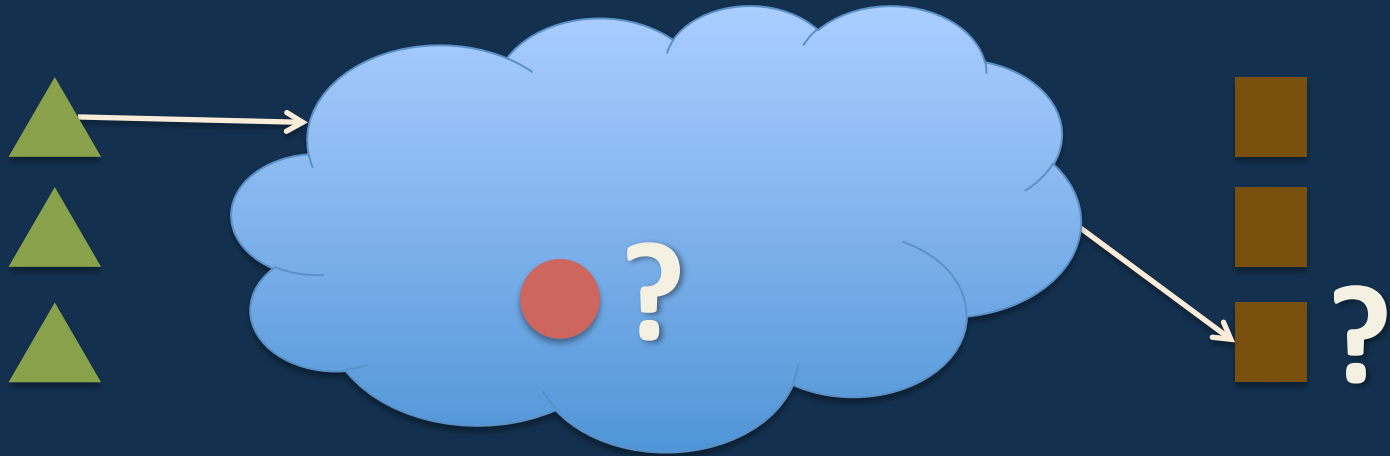
- Onion routing provides anonymous communication.

Overview



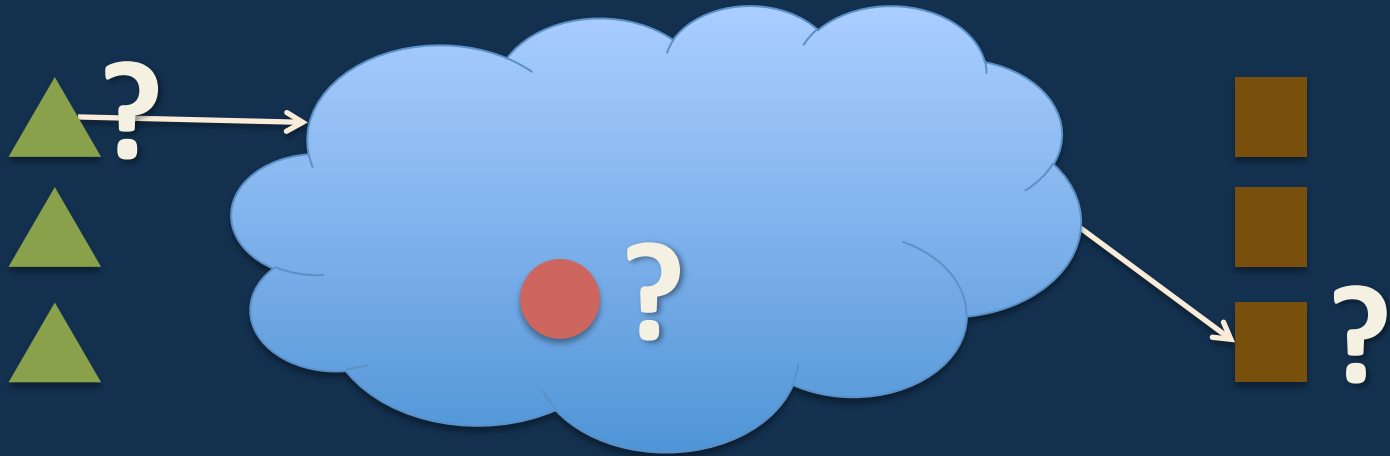
- Onion routing provides anonymous communication.

Overview



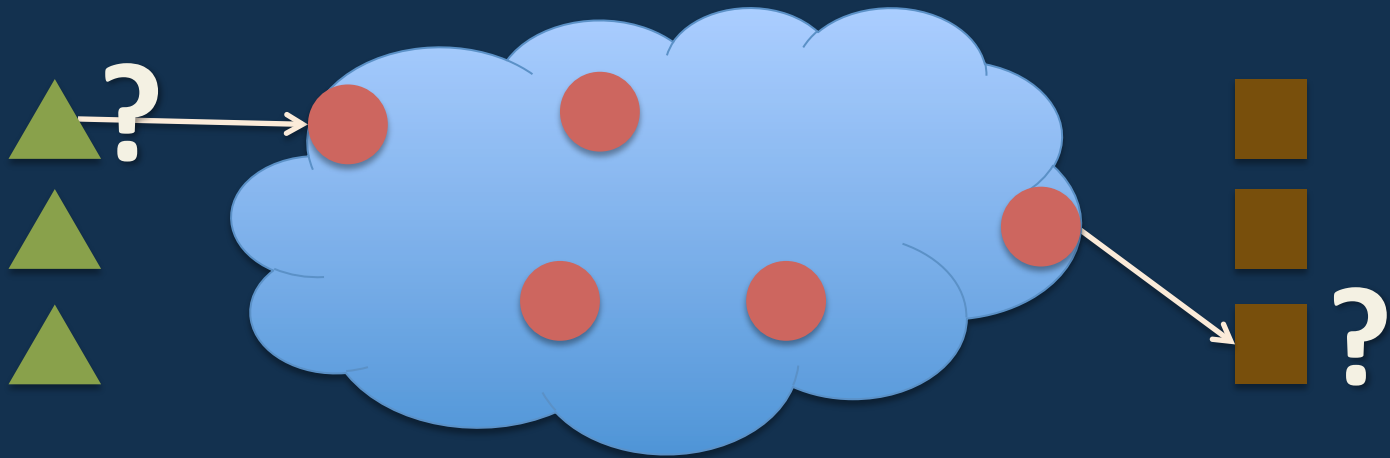
- Onion routing provides anonymous communication.

Overview



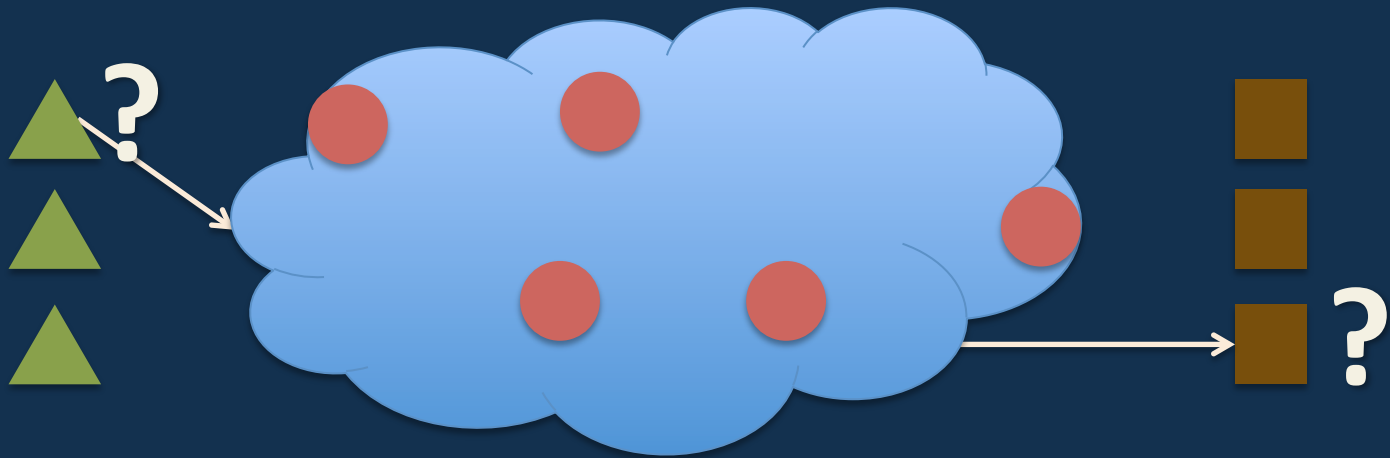
- Onion routing provides anonymous communication.

Overview



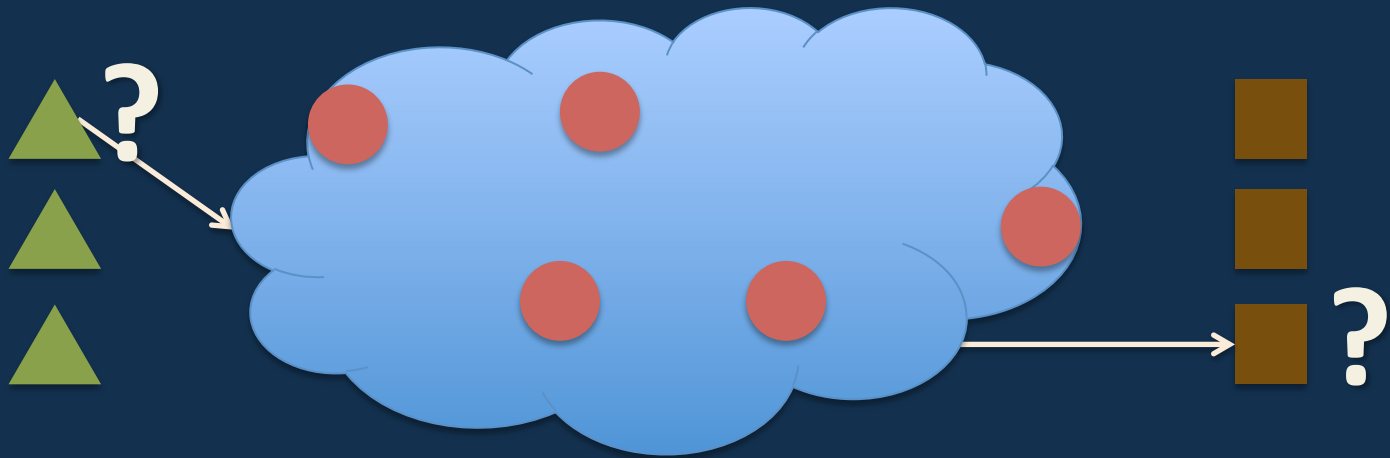
- Onion routing provides anonymous communication.
- It is insecure against an adversary with resources.

Overview



- Onion routing provides anonymous communication.
- It is insecure against an adversary with resources.
- Trust can help avoid such an adversary.
 - We provide a model of trust.
 - We design trust-based routing algorithms.

Overview

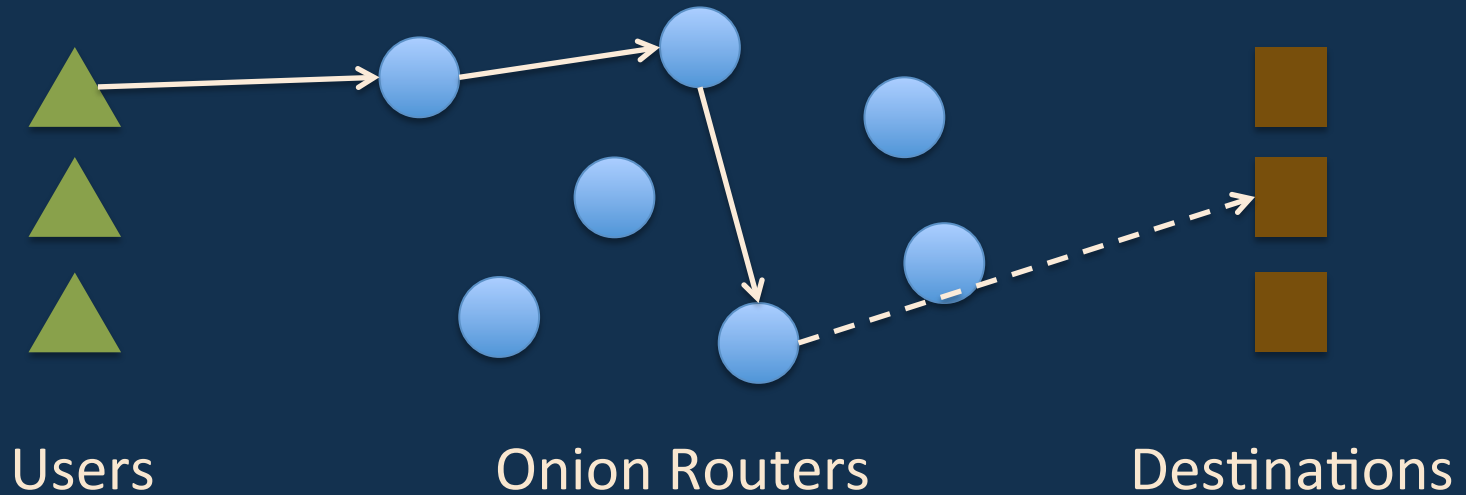


- Onion routing provides anonymous communication.
- It is insecure against an adversary with resources.
- Trust can help avoid such an adversary.
 - We provide a model of trust.
 - We design trust-based routing algorithms.
- Improve anonymity with robustness to trust errors.

Onion Routing

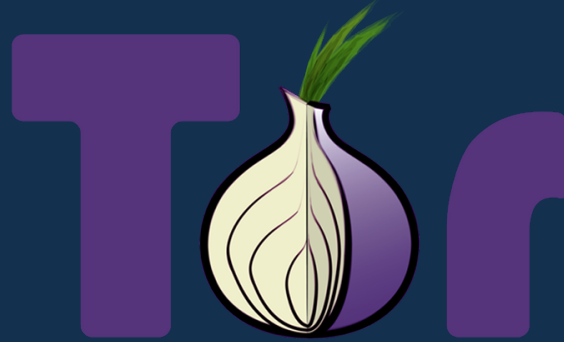


Onion Routing



1. A user cryptographically constructs a circuit through the network.
2. Onion-encrypted data is sent and unwrapped along the circuit.
3. The process runs in reverse for return data.

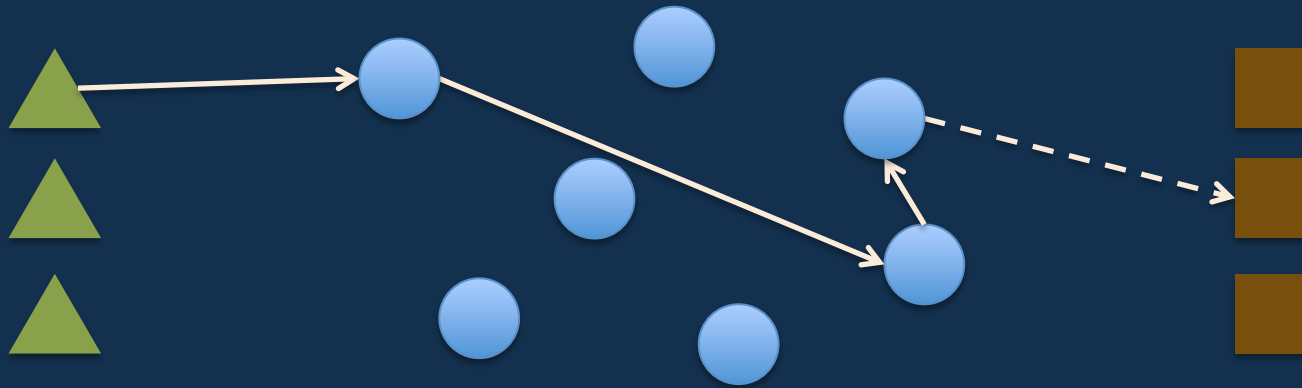
Onion Routing



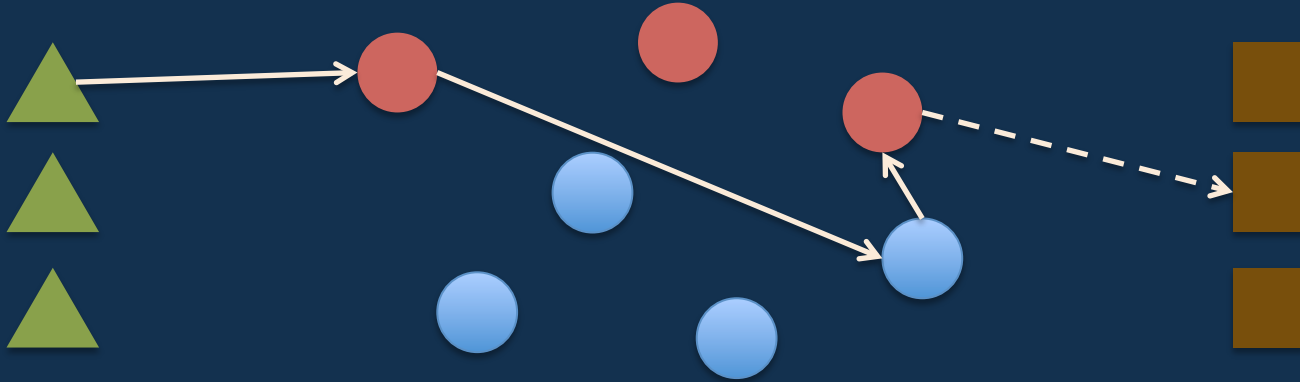
torproject.org

- International onion-routing network
- Estimated at over 300,000 users daily
- ≈2500 onion routers
- Uses include
 - Avoiding censorship
 - Gathering intelligence
 - Political activism
 - Whistleblowing

Problem

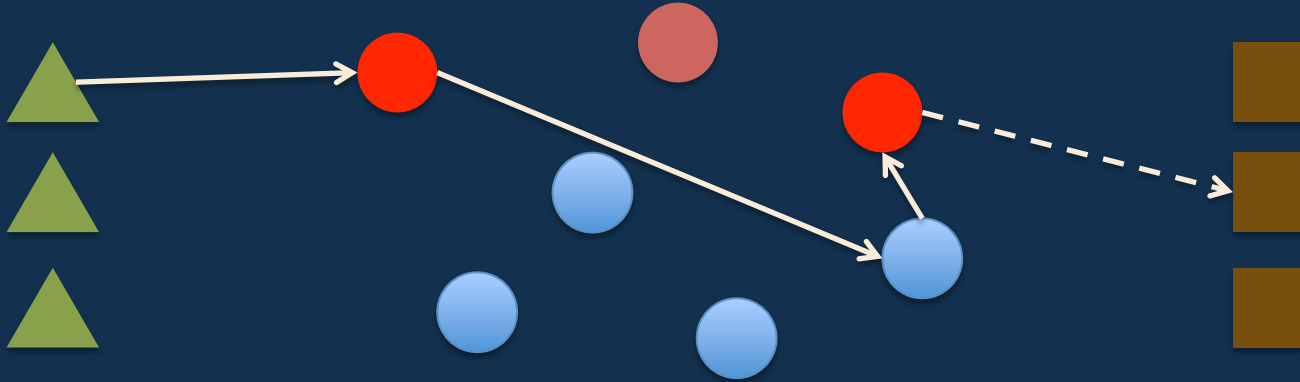


Problem



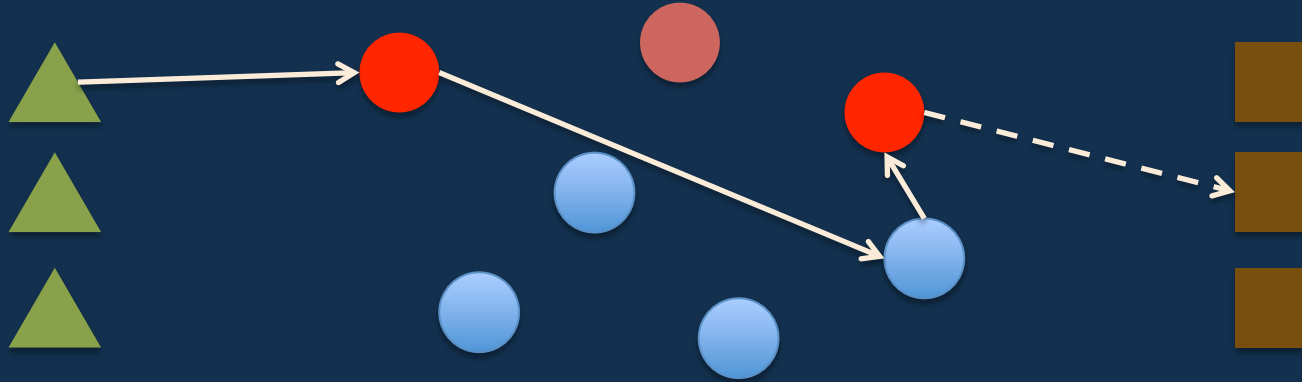
- Adversary may observe or control routers.

Problem



- Adversary may observe or control routers.
- Traffic patterns can link first and last routers.

Problem



- Adversary may observe or control routers.
- Traffic patterns can link first and last routers.

First-last Correlation Attack Success

Path Selection	Probability of attack success	# routers observed	# connections until successful attack
Random	0.01	250	100
+ guards & exits	0.01	80 guards, 90 exits	10 w/ prob. 0.1
+ bandwidth weighting	0.01	guard&exit, 124 MiBps	7.7 w/ prob. 0.077

2500 total routers, 900 exit, 800 guard

Key Idea: Trust

- Users may know how likely a router is to be under observation.

Tor Routers with Possible Trust Factors




Name	Hostname	Bandwidth	Uptime	Location	Tor version	OS
moria1	moria.csail.mit.edu	460 KB/s	1 days	USA	0.2.3.5-alpha	Linux
rathergonaked	212-82-33-112.ip.14v.de	302 KB/s	6 days	Germany	0.2.2.33	Linux
Unnamed	static-ip-166-154-142-114.rev.dyxnet.com	58 KB/s	58 days	Hong Kong	0.2.1.29	Windows Server 2003 SP2

Source: <http://torstatus.blutmagie.de>, 10/12/2011

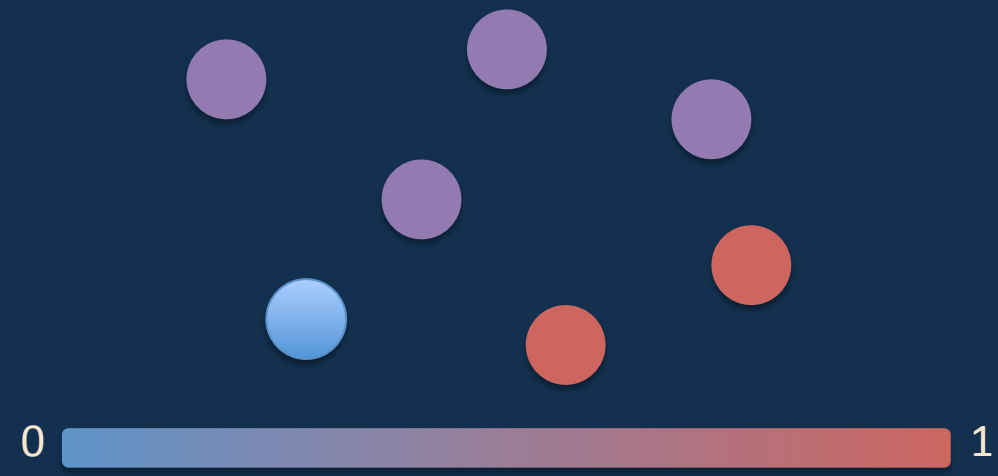
Problems

1. What is trust?
 - Model
2. How do we use trust?
 - Path-selection algorithm

Model

- User u 
- Naïve 
- users N 

Observed source



- Observed destination 
- 
- 

Probability of Compromise: $c^u(r)$
Trust: $\tau^u(r) = 1 - c^u(r)$

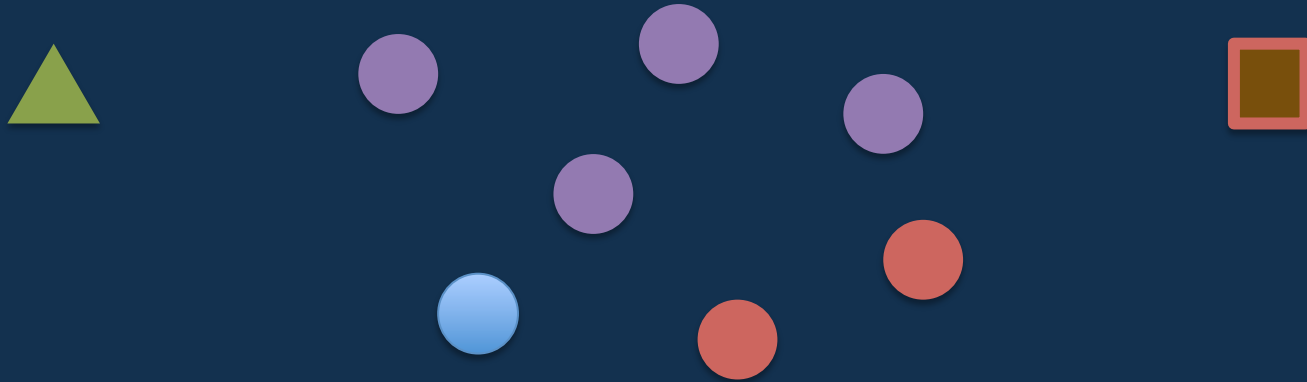
- A^u 
 - 
 - 
- Adversaries

Trust-based Path Selection Algorithm

1. Destination links observed only
 - Use *downhill algorithm*.
2. Source links observed only
 - Use one-hop path of most-trusted router.
3. Neither source nor destination links observed
 - Connect directly to destination.
4. Both source and destination links observed
 - Connect directly to destination.

Downhill Algorithm

Key idea: Blend in with the naïve users.

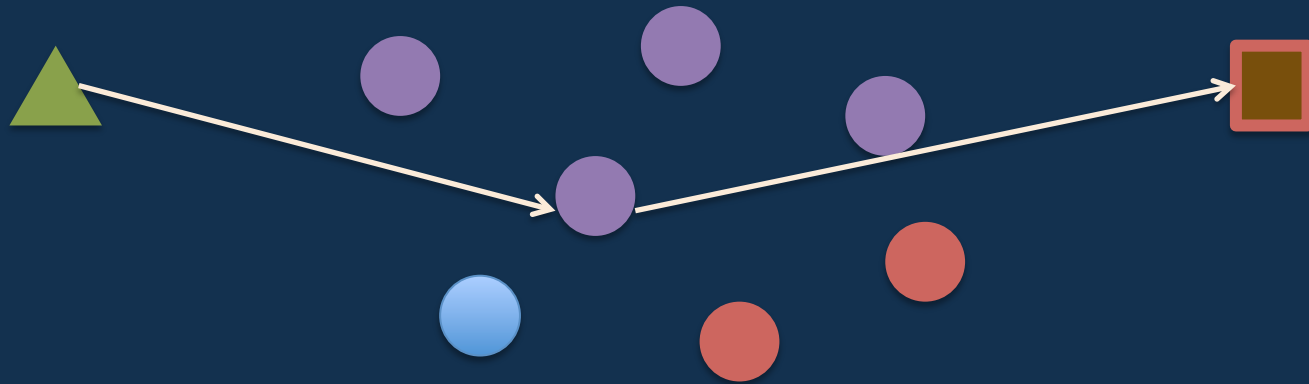


Random

Most trusted

Downhill Algorithm

Key idea: Blend in with the naïve users.

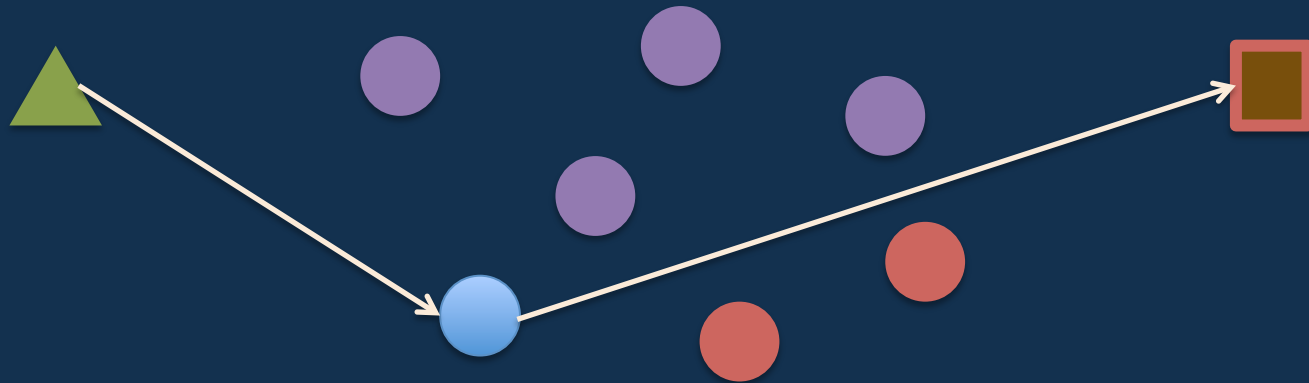


Random

Most trusted

Downhill Algorithm

Key idea: Blend in with the naïve users.

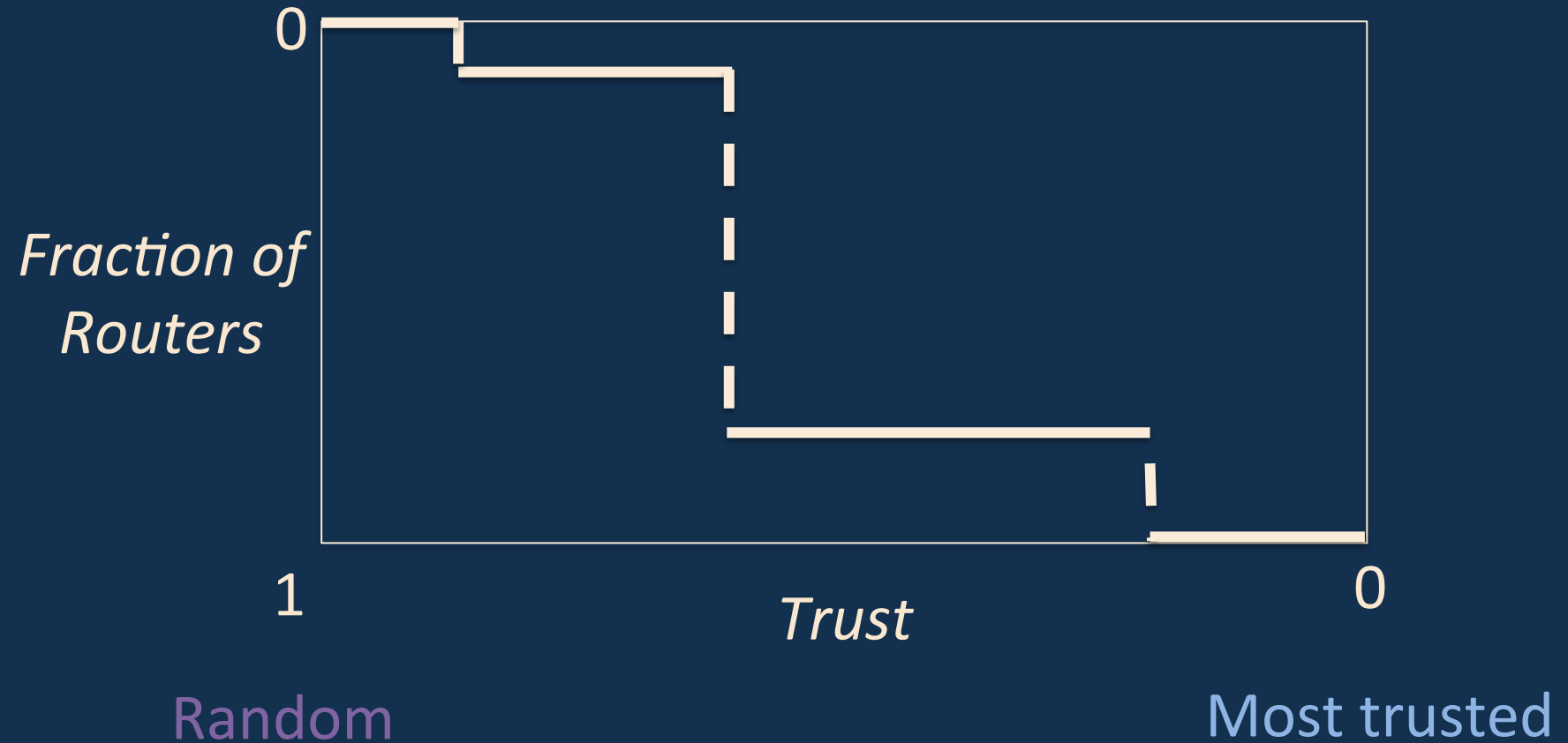


Random

Most trusted

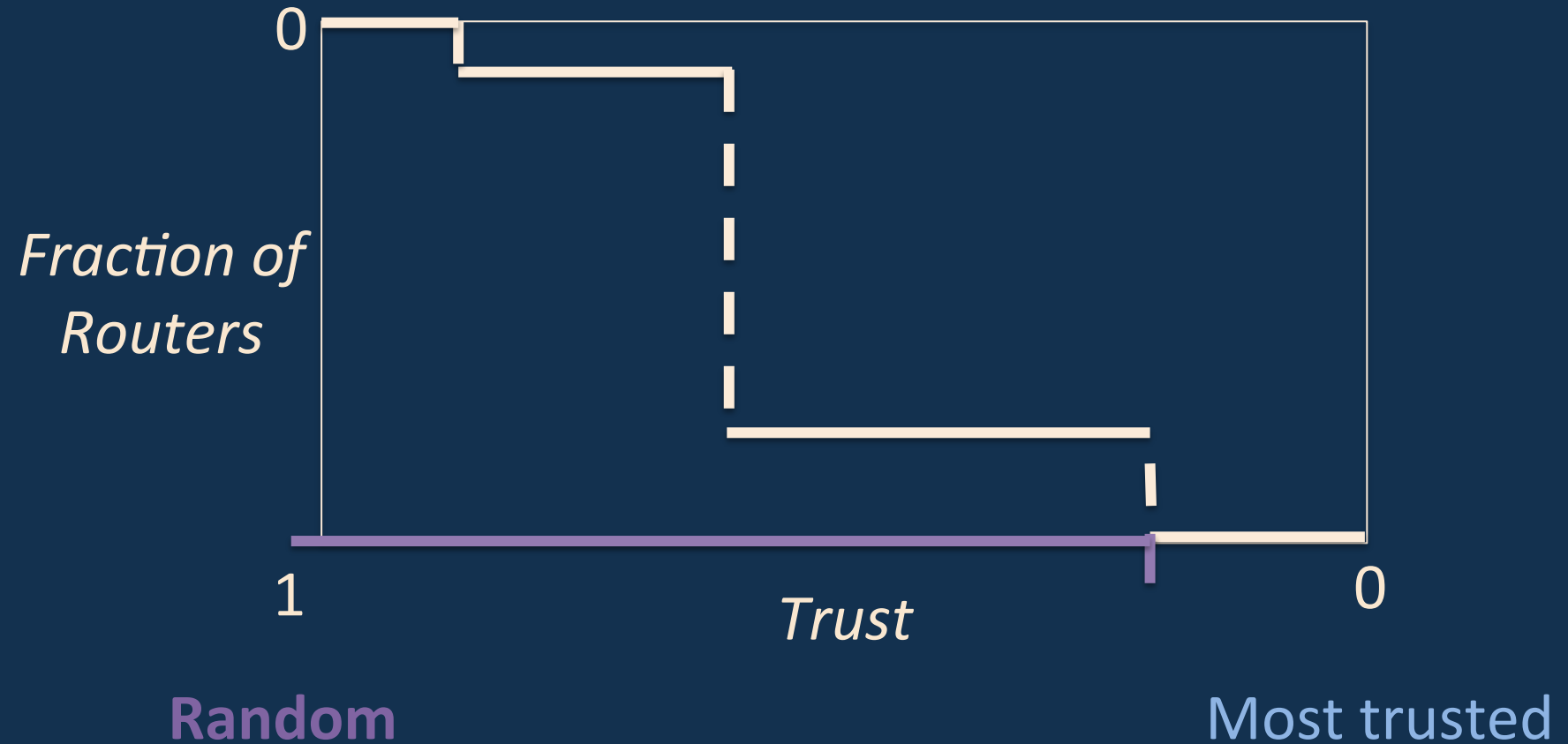
Downhill Algorithm

Router Trust CDF



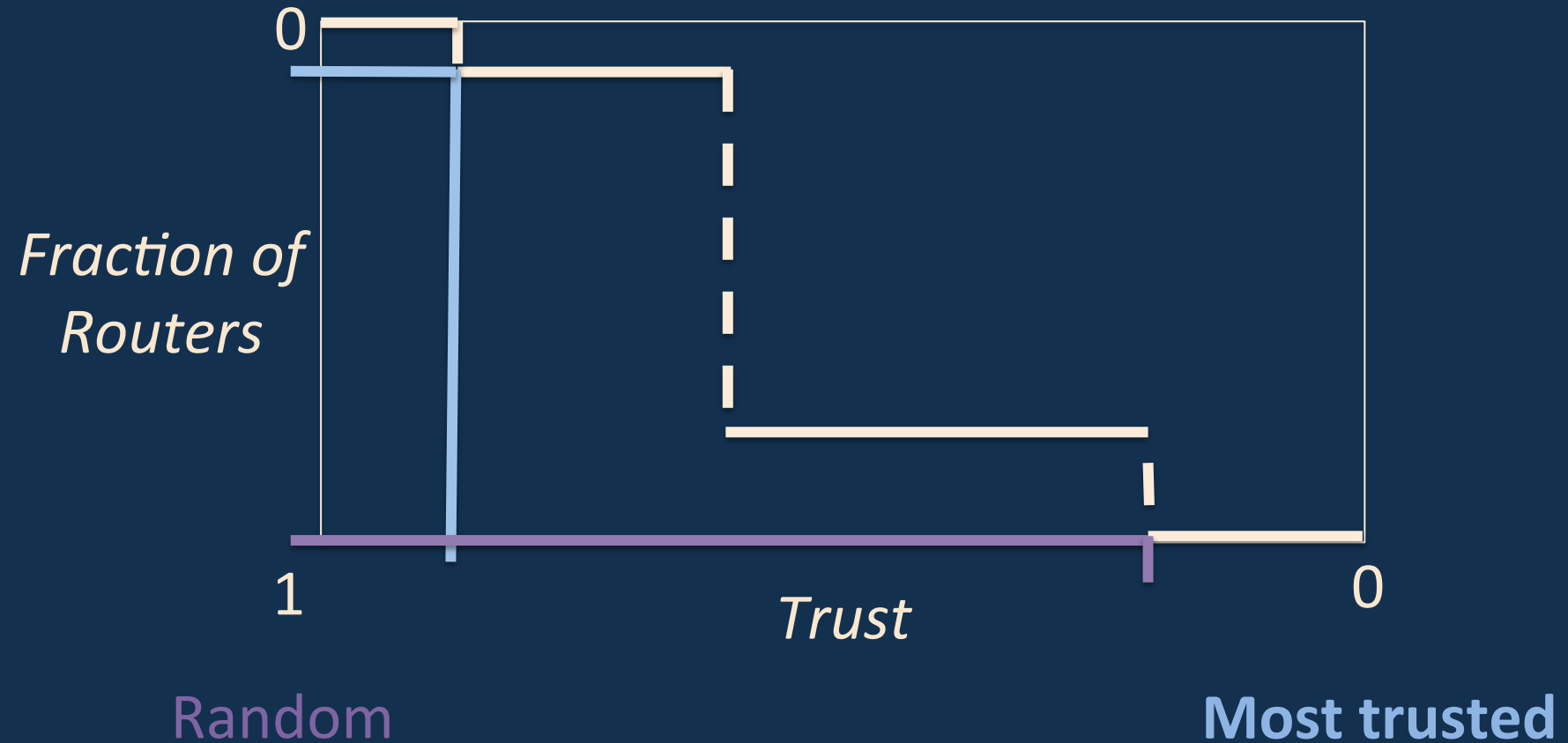
Downhill Algorithm

Router Trust CDF



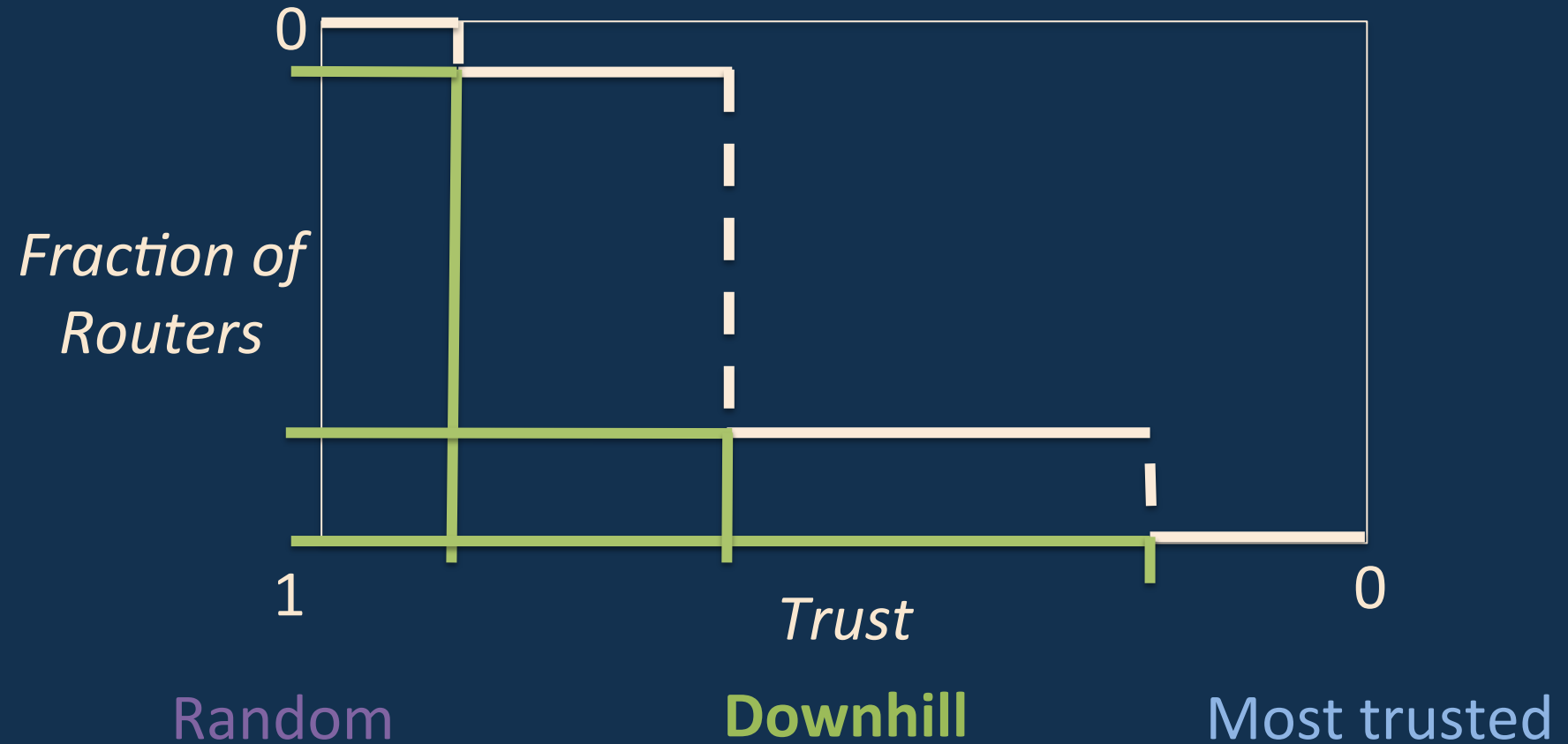
Downhill Algorithm

Router Trust CDF

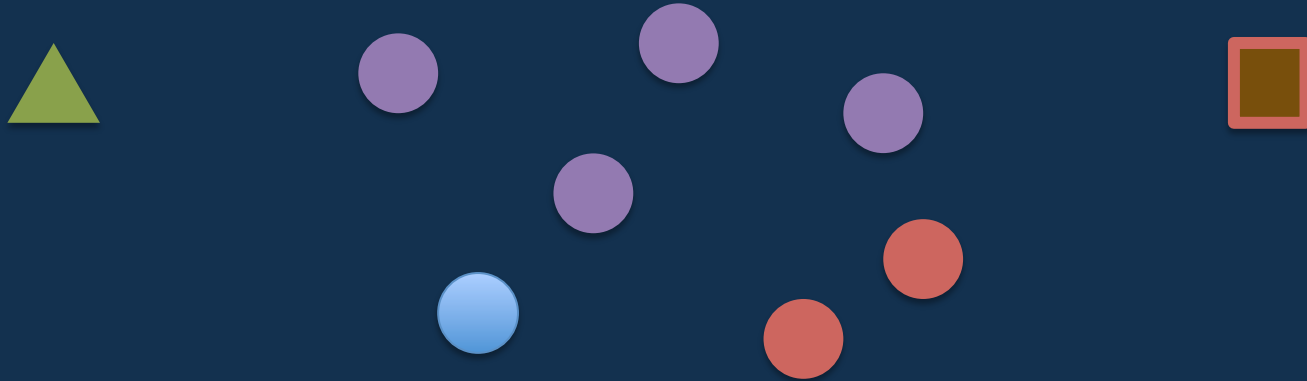


Downhill Algorithm

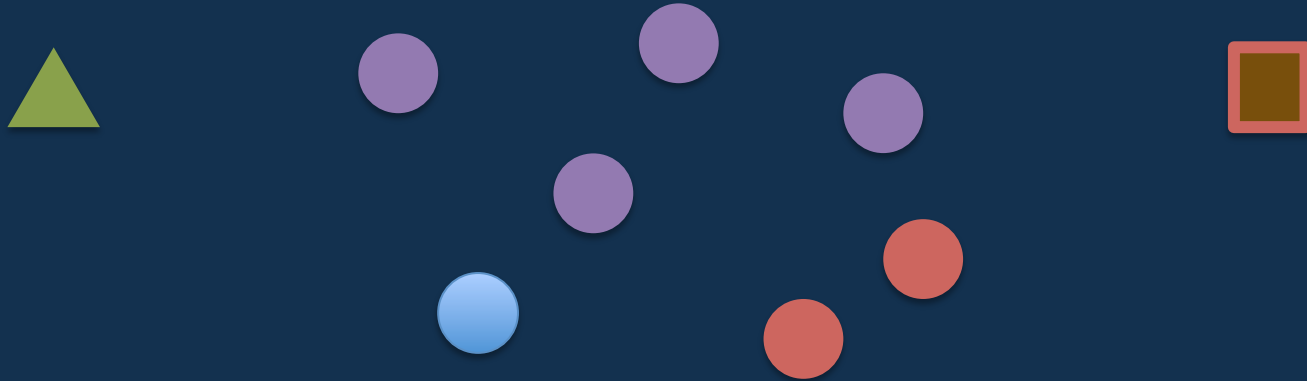
Router Trust CDF



Downhill Algorithm

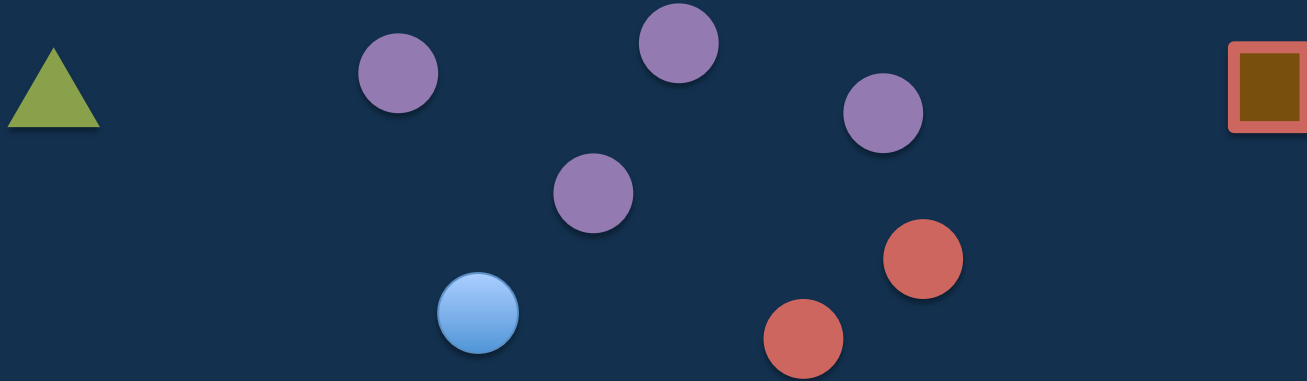


Downhill Algorithm



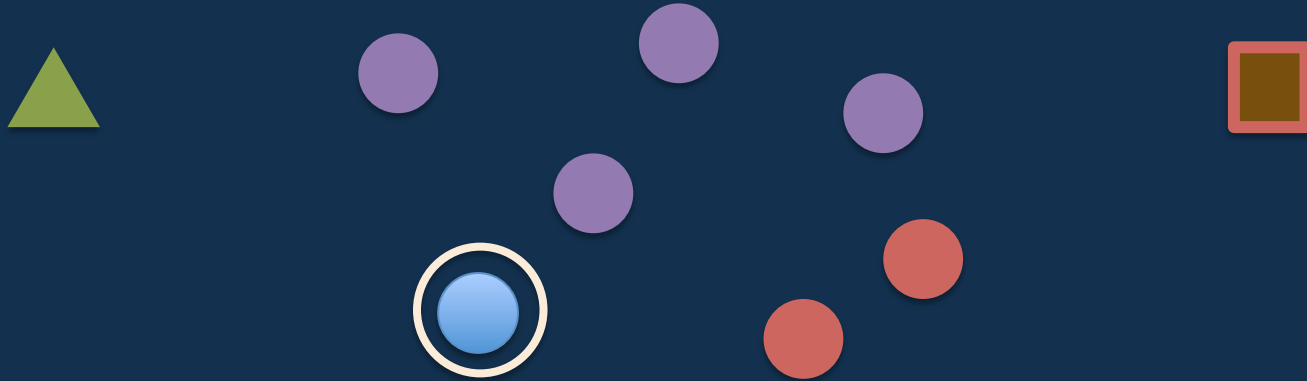
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.

Downhill Algorithm



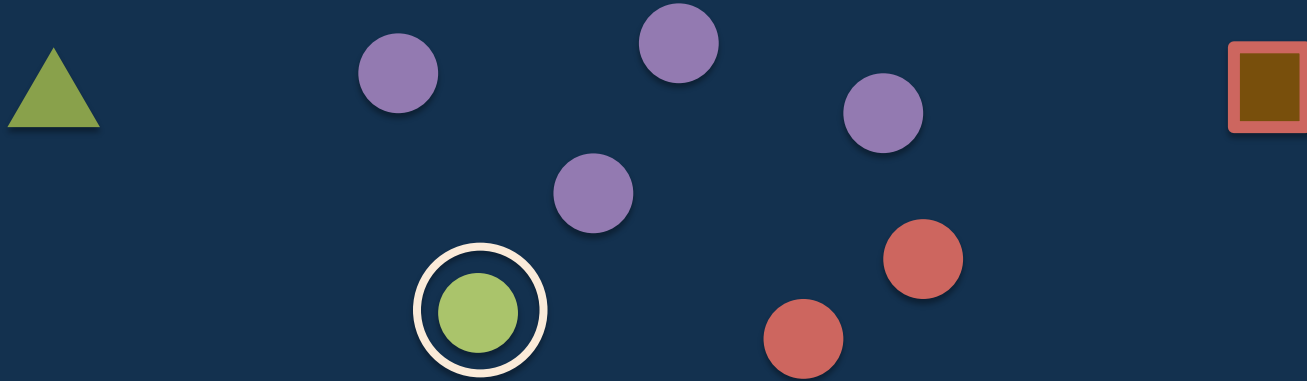
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



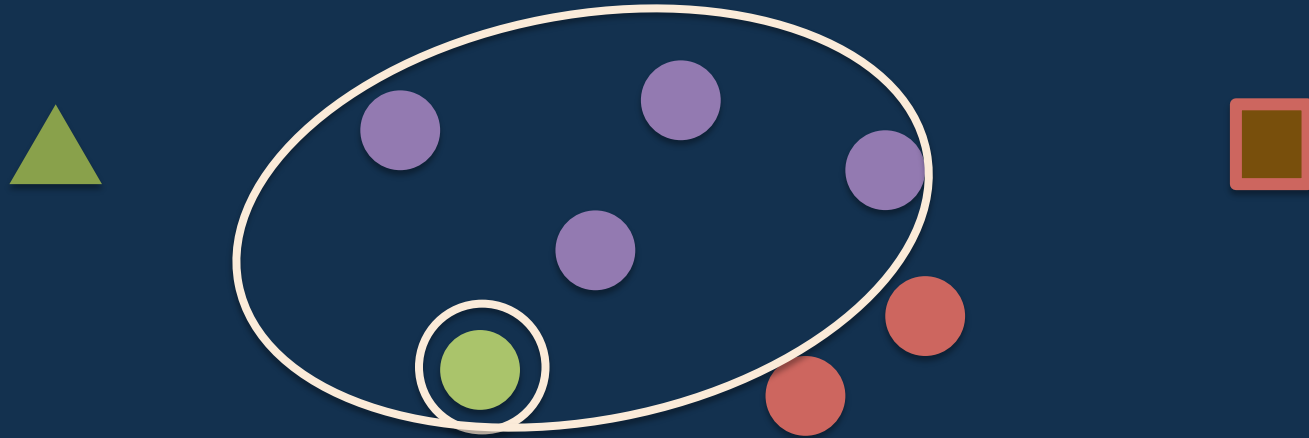
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



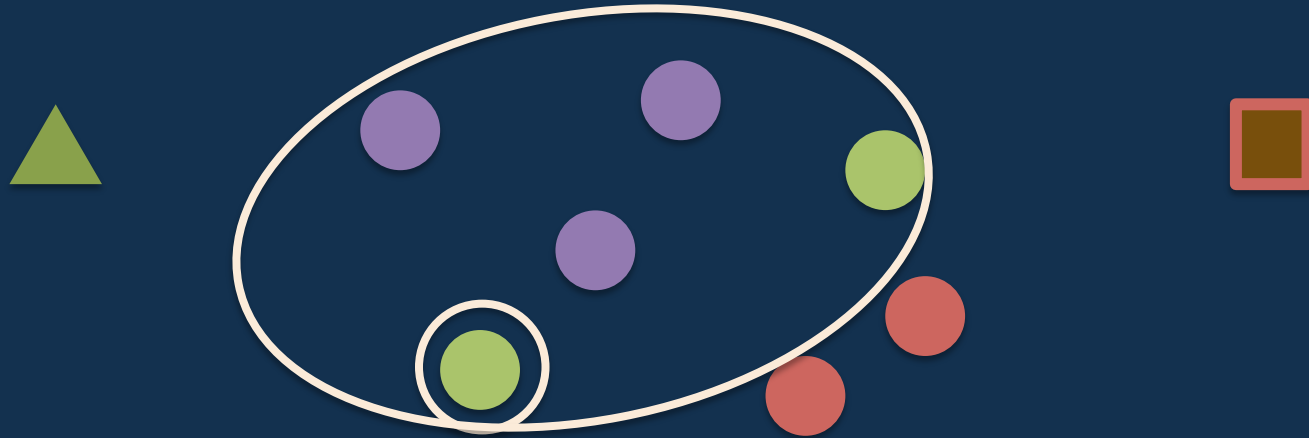
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



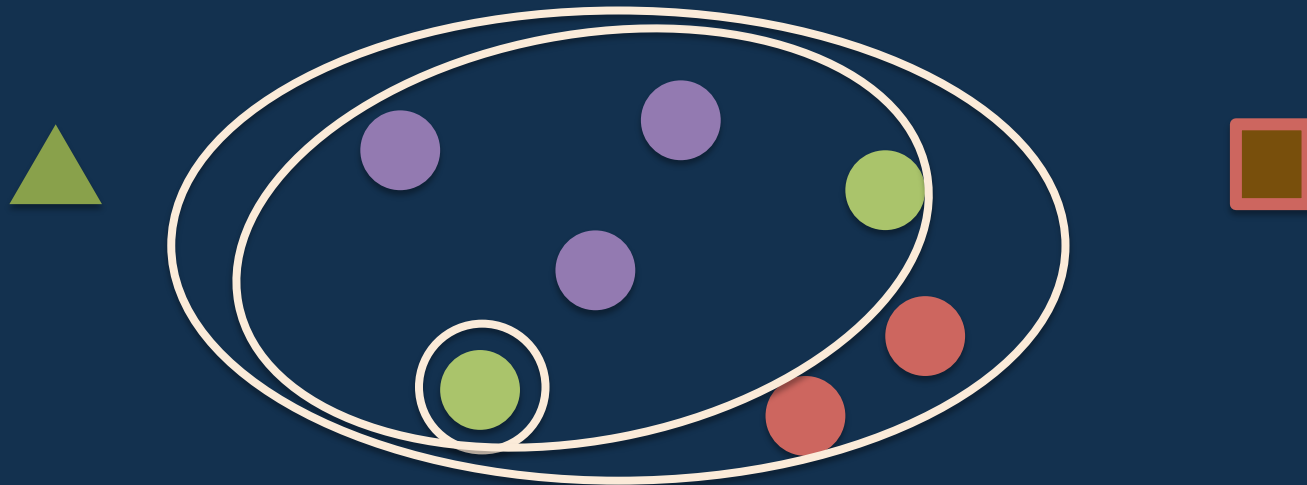
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



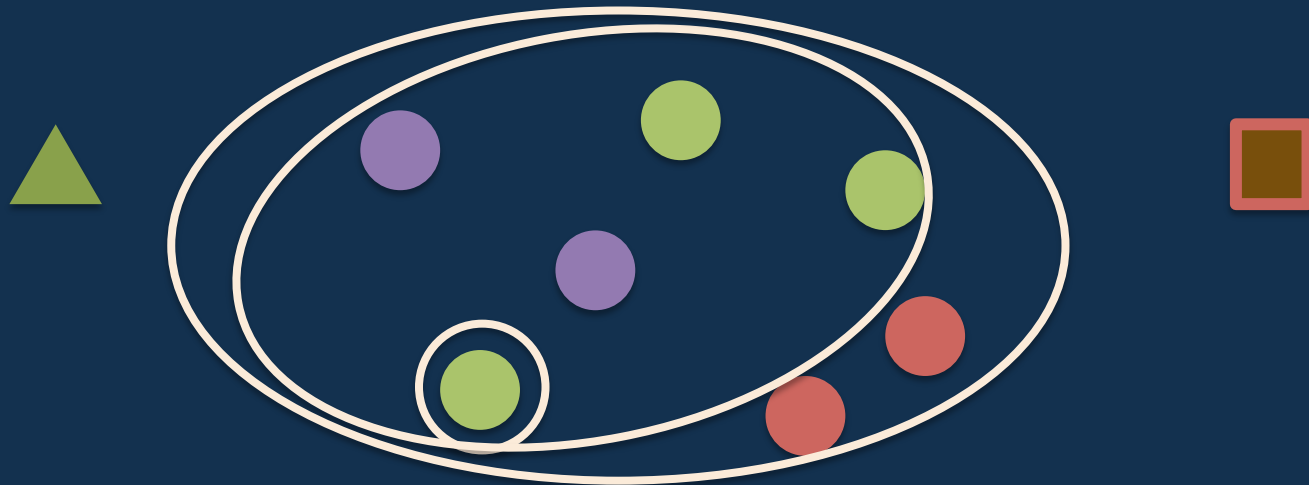
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



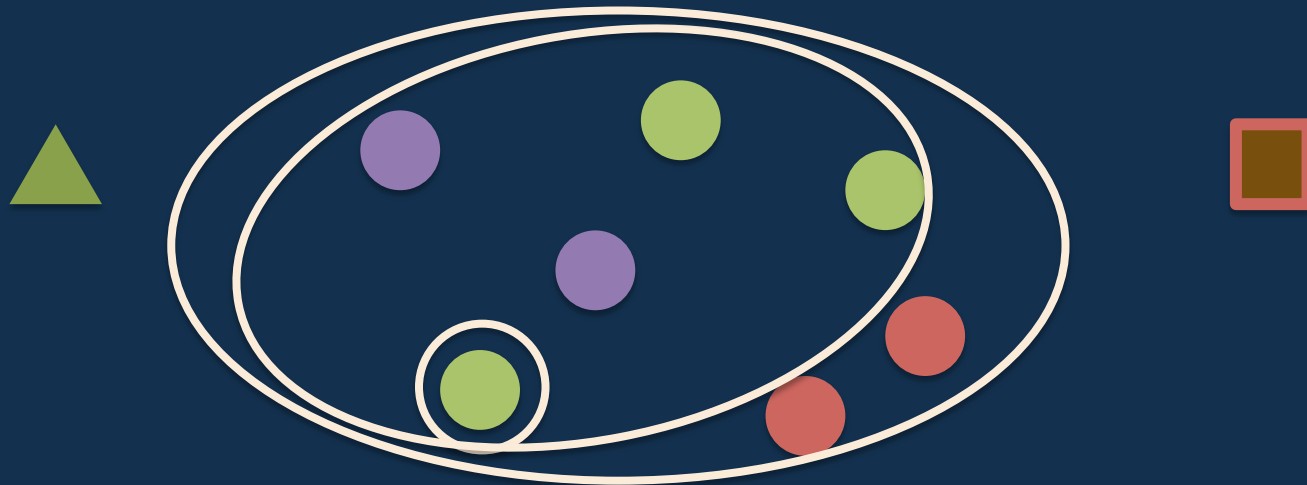
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



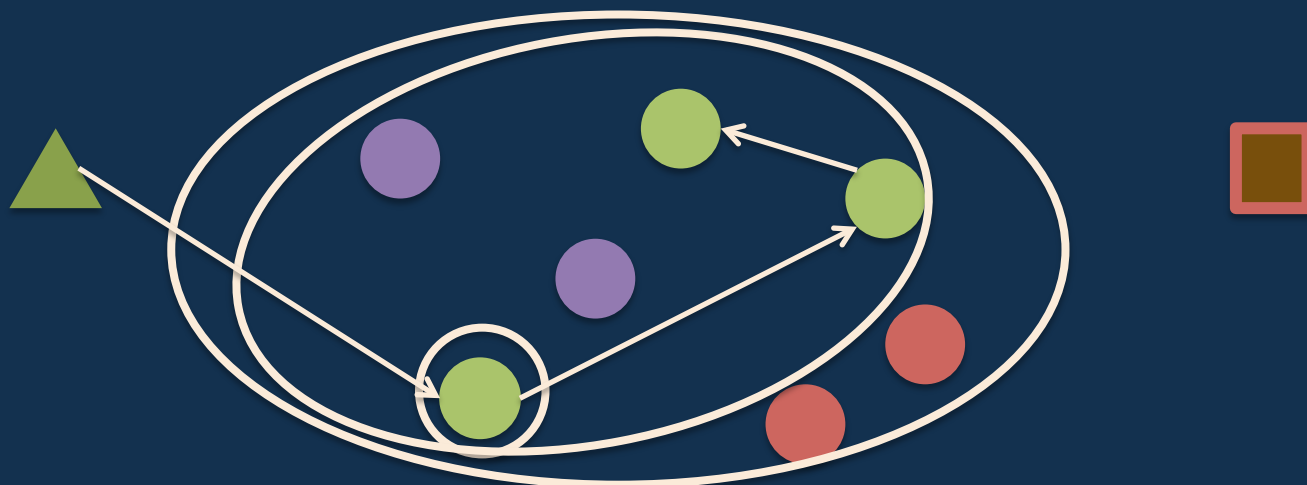
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$

Downhill Algorithm



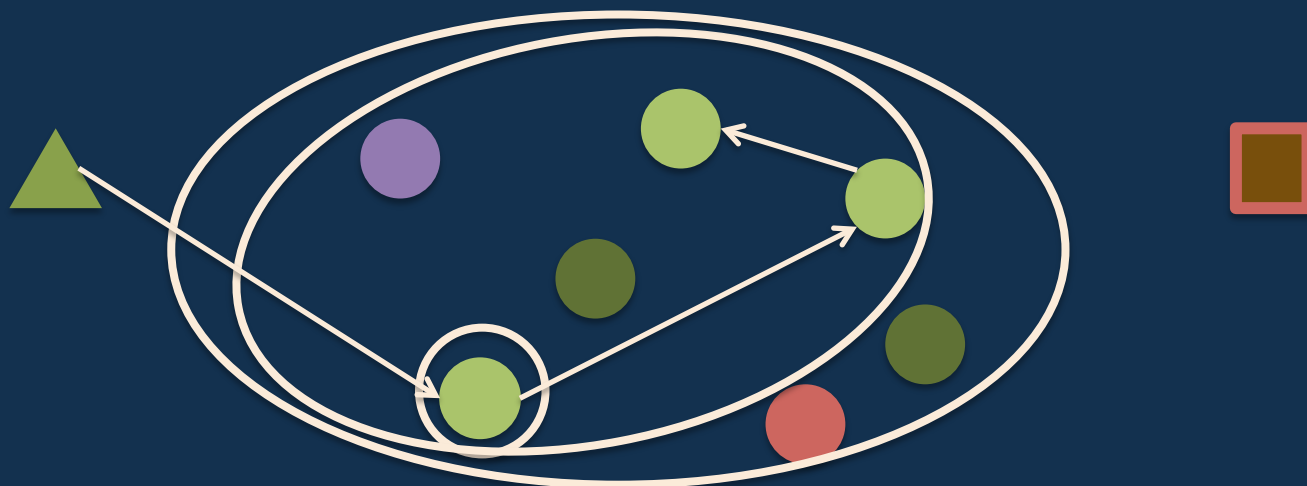
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$
3. For each connection,
 1. Create circuit through selected routers.
 2. Randomly choose two routers.
 3. Extend circuit through them to the destination.

Downhill Algorithm



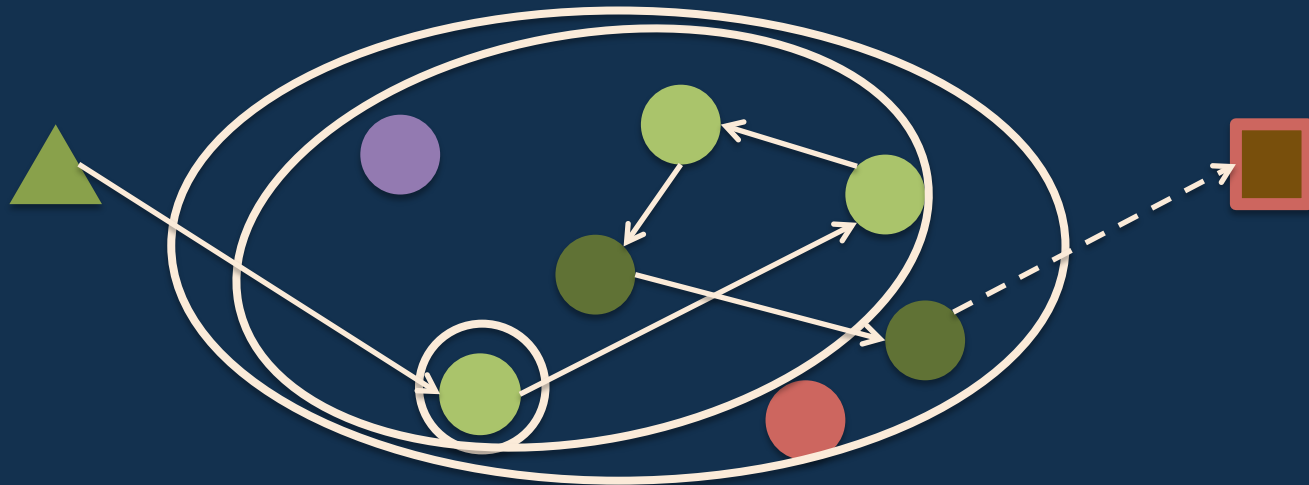
1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$
3. For each connection,
 1. Create circuit through selected routers.
 2. Randomly choose two routers.
 3. Extend circuit through them to the destination.

Downhill Algorithm



1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$
3. For each connection,
 1. Create circuit through selected routers.
 2. Randomly choose two routers.
 3. Extend circuit through them to the destination.

Downhill Algorithm

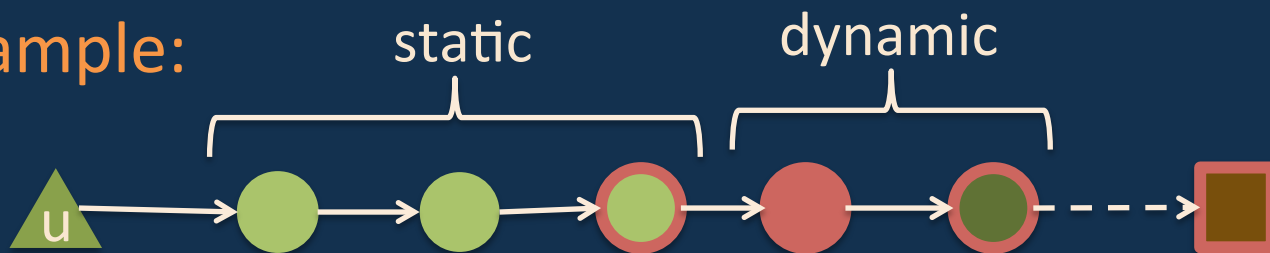


1. Set path length l and trust levels $\lambda_1, \dots, \lambda_l$ to optimize expectation of anonymity metric.
2. For $1 \leq i \leq l$,
Randomly select among routers with trust $\geq \lambda_i$
3. For each connection,
 1. Create circuit through selected routers.
 2. Randomly choose two routers.
 3. Extend circuit through them to the destination.

Anonymity Analysis

- Metric: Posterior probability of actual source of a given connection.

Example:



- Let $T_i = \{r : \tau_u(r) \geq \lambda_i\}$.
- Let $A^u \subset R$ be the routers compromised by A^u .
- Let $X_1 = \Pr[u \text{ chose observed path}]$
$$= (|T_1 \setminus A^u| / |T_1|) (|T_2 \setminus A^u| / |T_2|) (1 / |T_3|) (1 / |R|)^2$$
- Let $X_2 = \Pr[n \in N \text{ chose observed path}]$
$$= (|R \setminus A^u| / |R|)^2 (1 / |R|)^3$$
- Posterior probability: $X_1 / (X_1 + |N| X_2)$

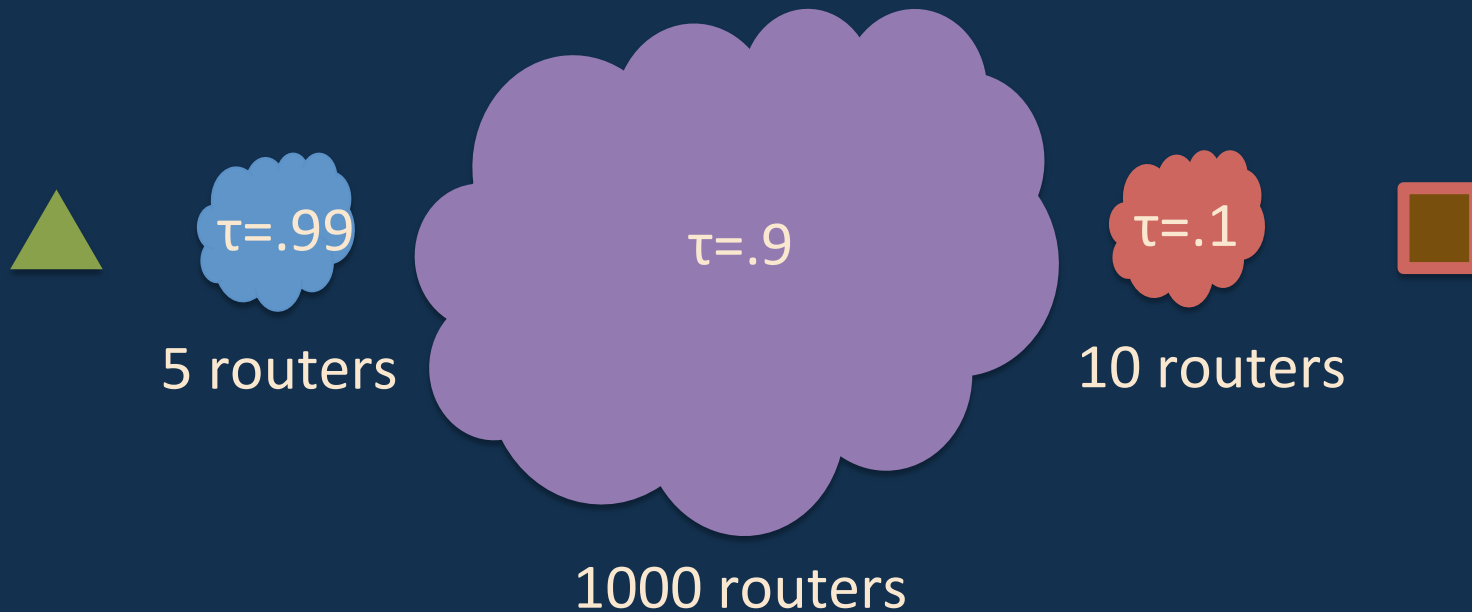
Anonymity Analysis

Expected anonymity	Downhill	Most trusted	Random	Lower bound
Many @ medium trust	0.0274	0.2519	0.1088	0.01
Many @ low trust	0.0550	0.1751	0.4763	0.001

Anonymity Analysis

Expected anonymity	Downhill	Most trusted	Random	Lower bound
Many @ medium trust	0.0274	0.2519	0.1088	0.01
Many @ low trust	0.0550	0.1751	0.4763	0.001

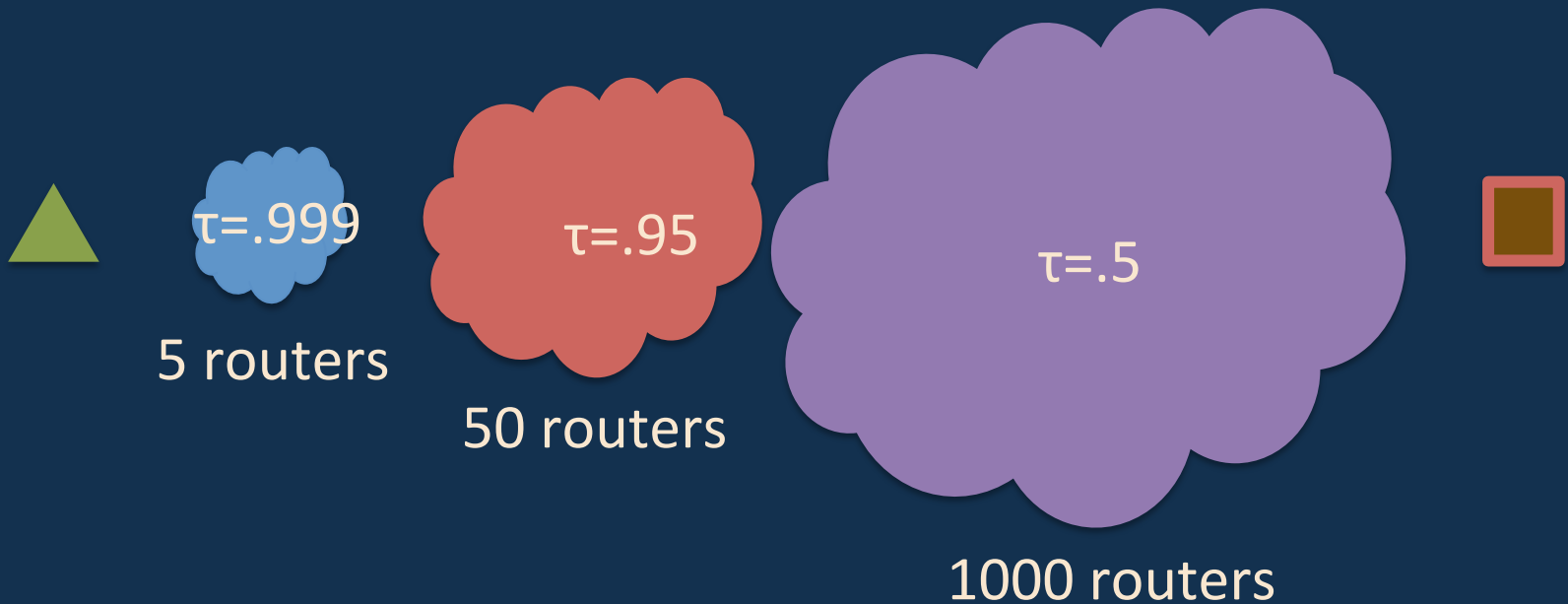
Scenario 1: User has some limited information.



Anonymity Analysis

Expected anonymity	Downhill	Most trusted	Random	Lower bound
Many @ medium trust	0.0274	0.2519	0.1088	0.01
Many @ low trust	0.0550	0.1751	0.4763	0.001

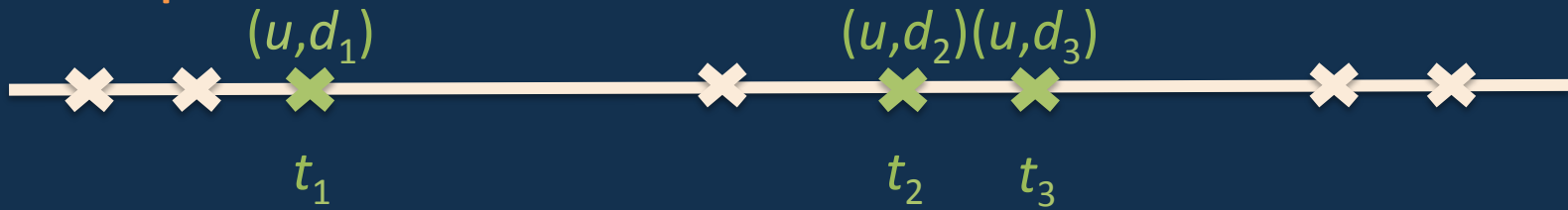
Scenario 2: User and friends run routers. Adversary is strong.



Linking Analysis

- Metric: Connection entropy at times user communicates.

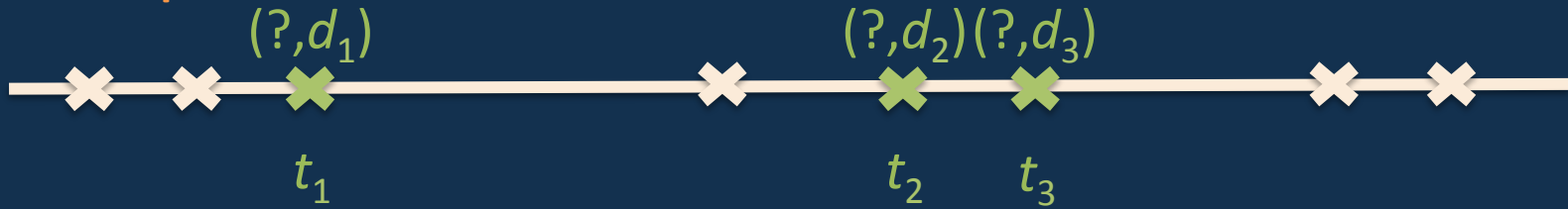
Example:



Linking Analysis

- Metric: Connection entropy at times user communicates.

Example:



The adversary may not know the user.

Linking Analysis

- Metric: Connection entropy at times user communicates.

Example:



Connections without dynamic hops may be linked by final hop.

Linking Analysis

- Metric: Connection entropy at times user communicates.

Example:

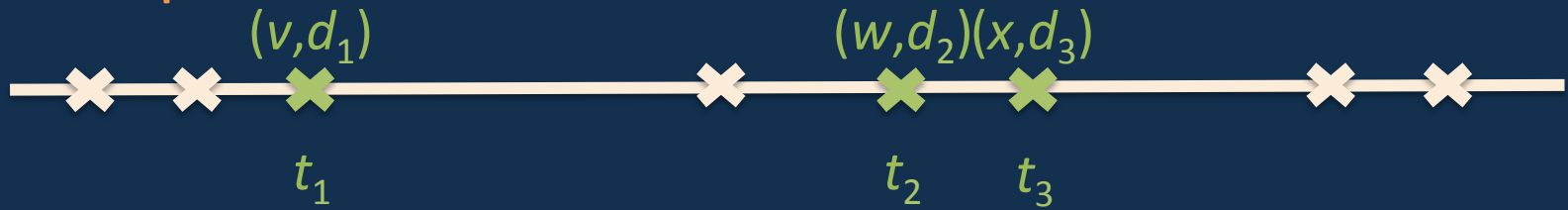


With dynamic hops, posterior distribution may be more even.

Linking Analysis

- Metric: Connection entropy at times user communicates.

Example:



With dynamic hops, posterior distribution may be more even.

Theorem:

Entropy of connection distribution is increased by using dynamic hops.

Trust Errors

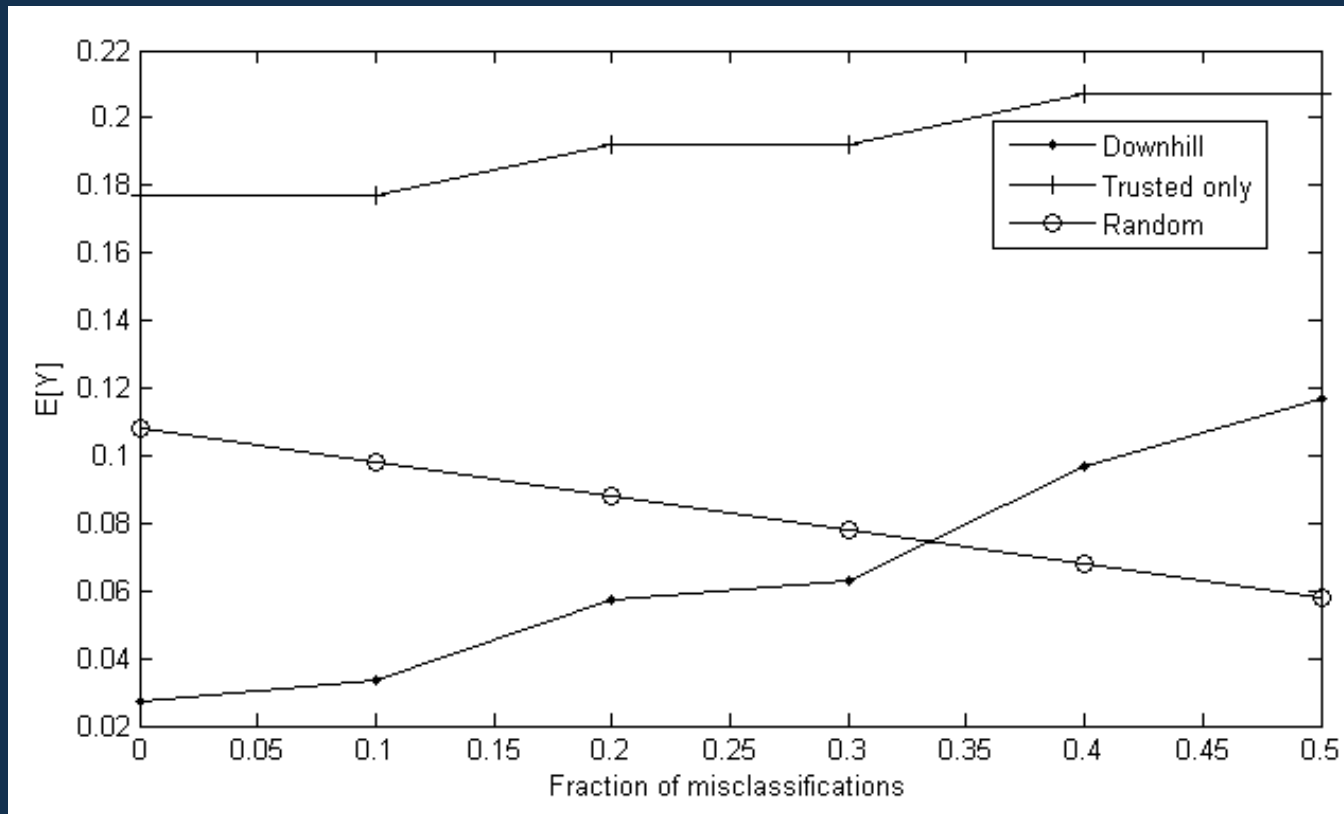
Theorem (informal):

Error in trust of router r changes expected anonymity proportional to

1. Size of error
2. Expected number of times r is used
3. Expected relative size of r 's trust set

Trust Errors

Errors in Scenario 1 (many @ medium trust)

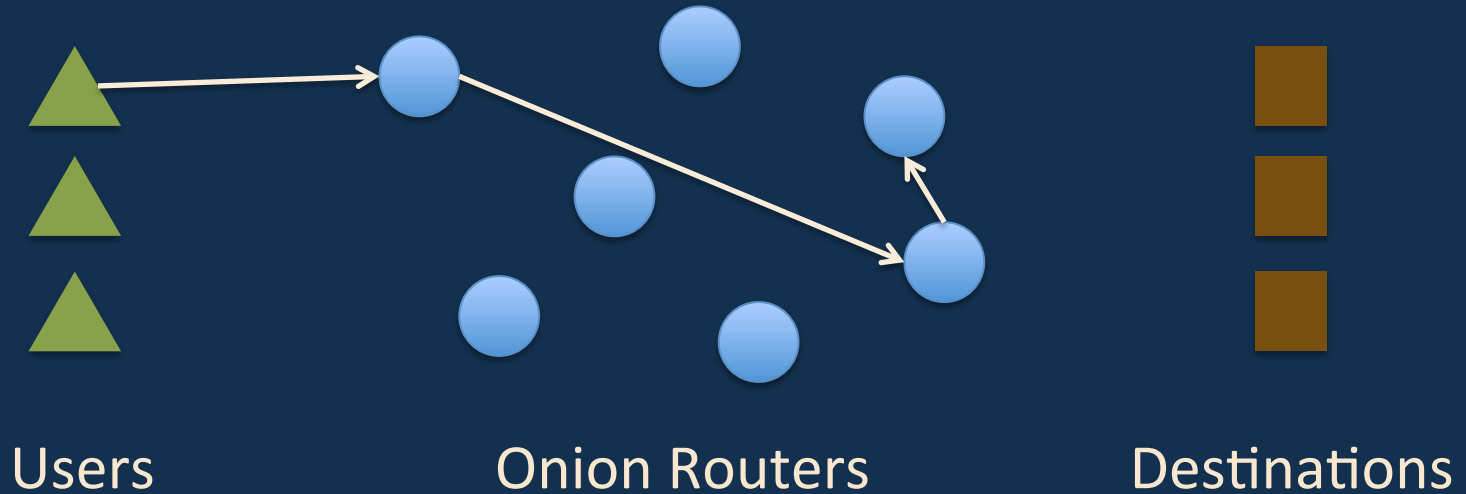


Fraction x of low are medium,
 $x/2$ of med. are low, $x/2$ of med. are high,
 x of high are medium.

Conclusion

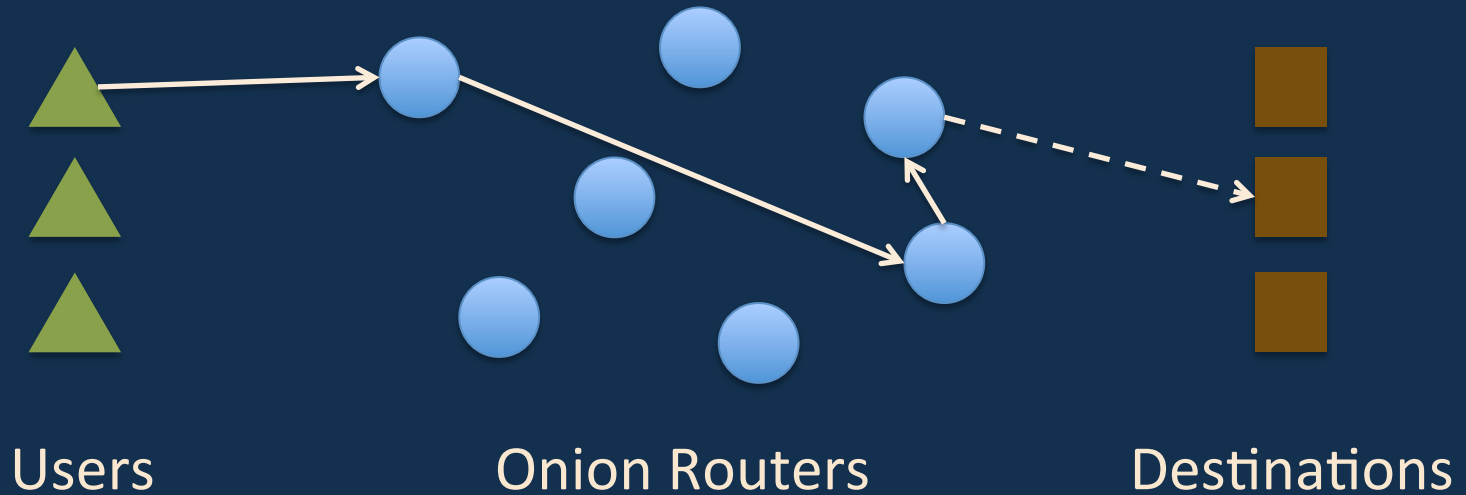
- Adversaries can attack onion-routing network like Tor today.
- External trust can provide protection.
 - We provide a model to express the problem and solution.
 - We give a path-selection algorithm and show that it improves anonymity.
- Future Work
 - Dependent compromise
 - Link adversary
 - Multiple adversaries per user
 - “Road warrior”
 - Private trust values
 - Private adversaries

Onion Routing



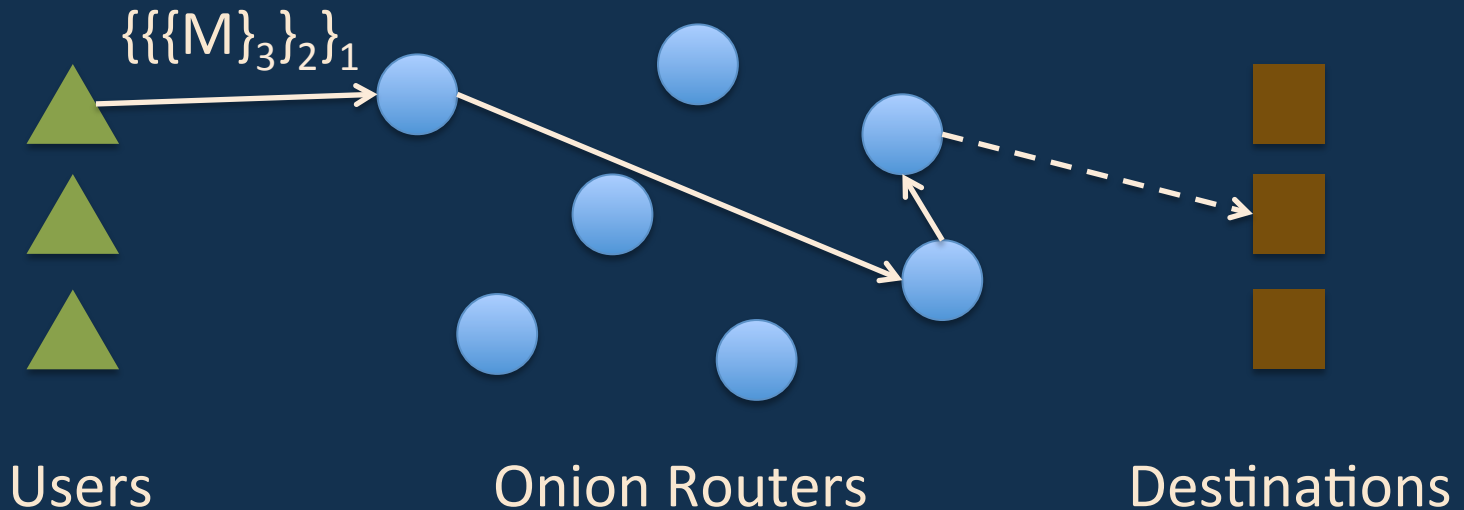
1. A user cryptographically constructs a circuit through the network.

Onion Routing



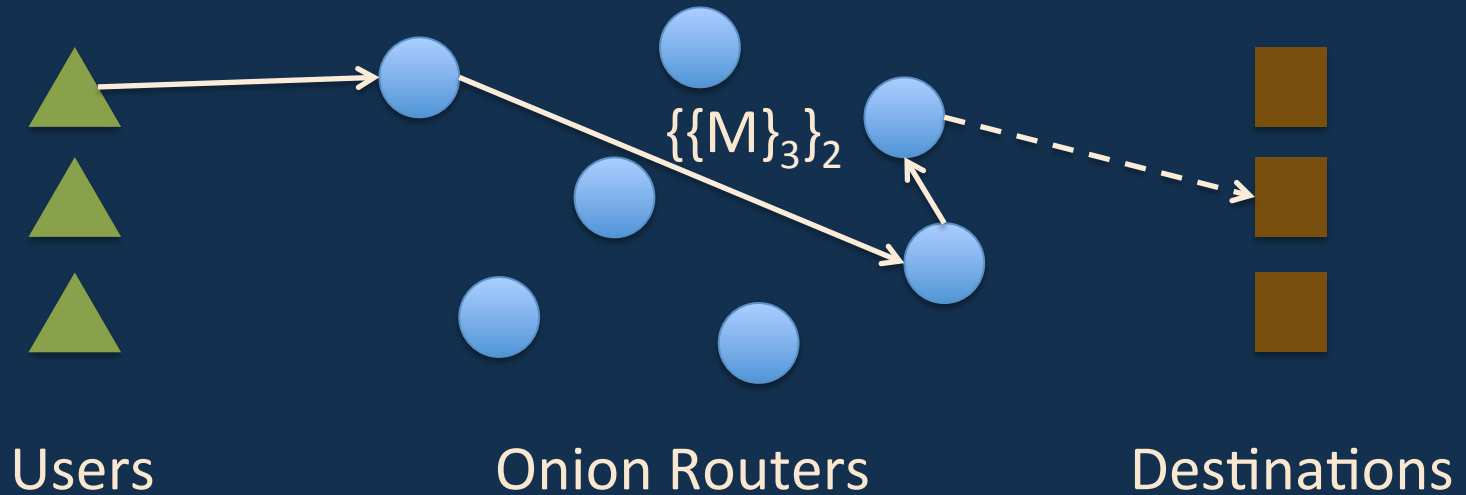
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.

Onion Routing



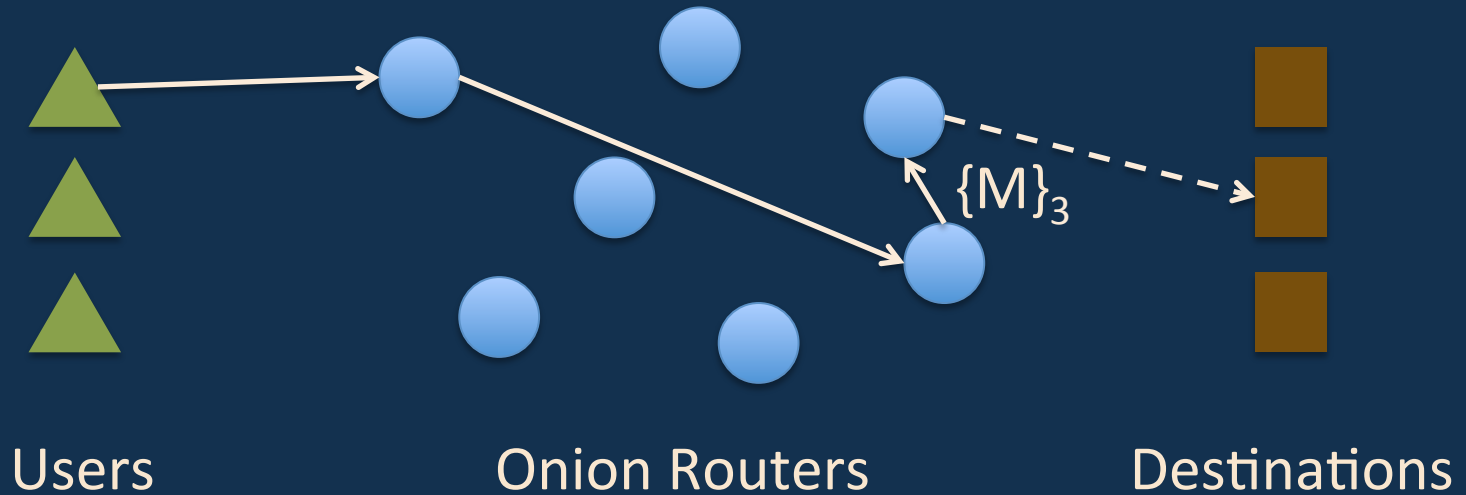
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.

Onion Routing



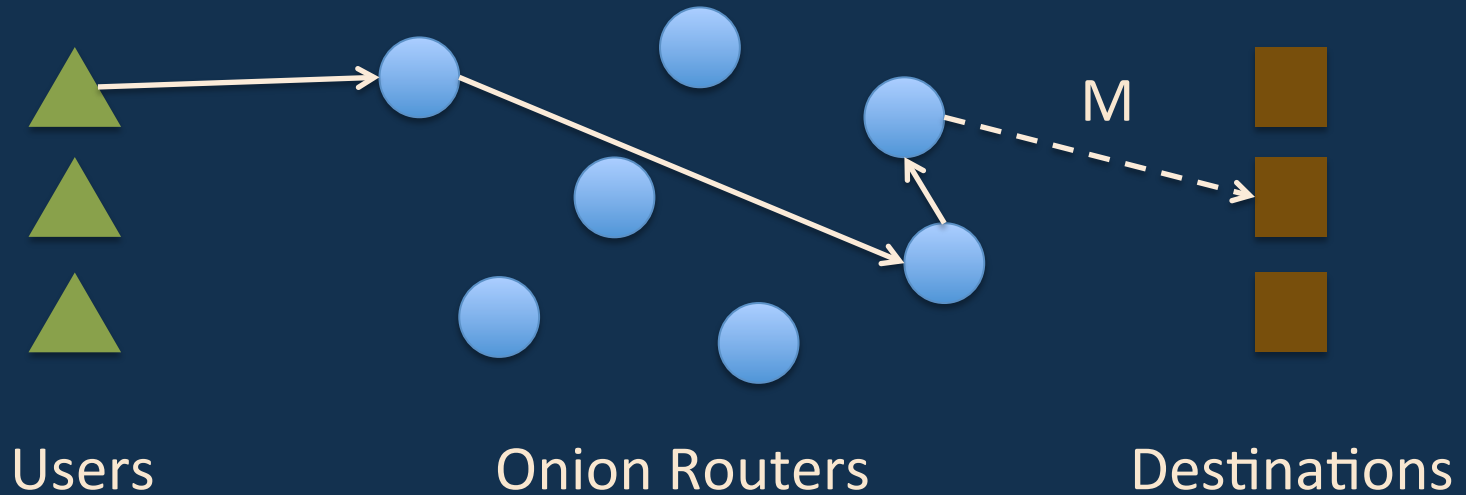
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.

Onion Routing



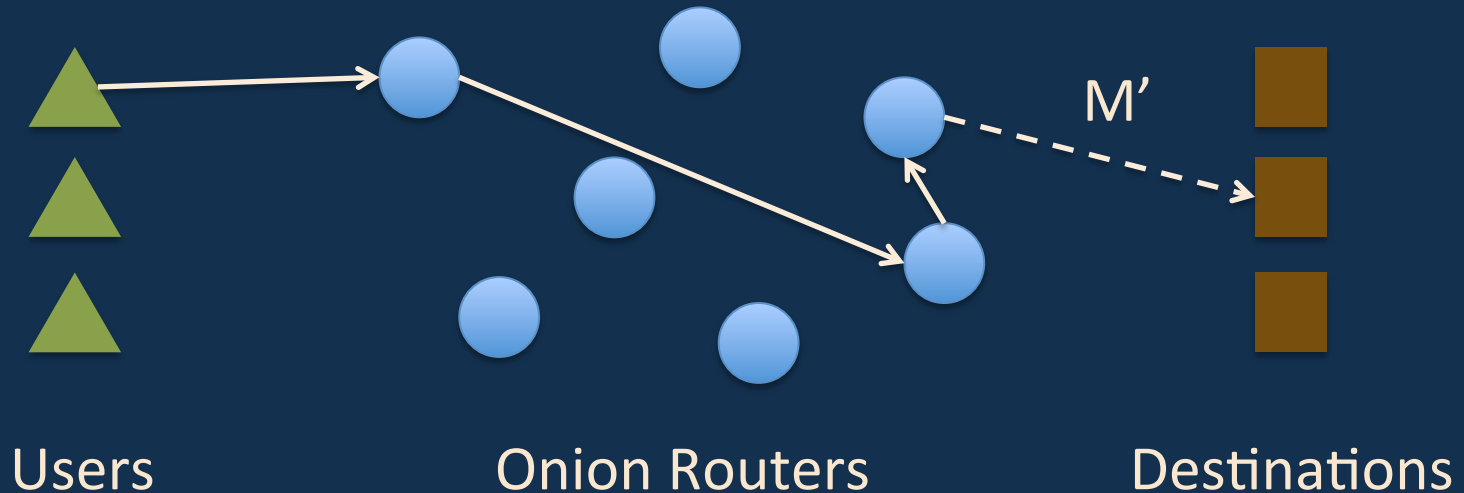
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.

Onion Routing



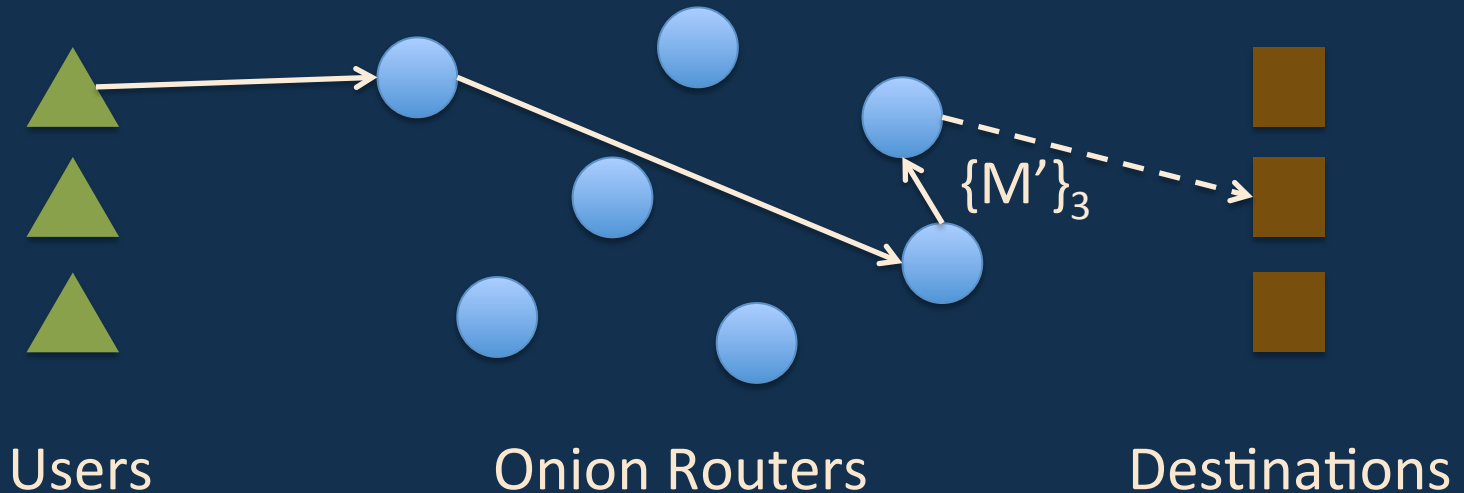
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.

Onion Routing



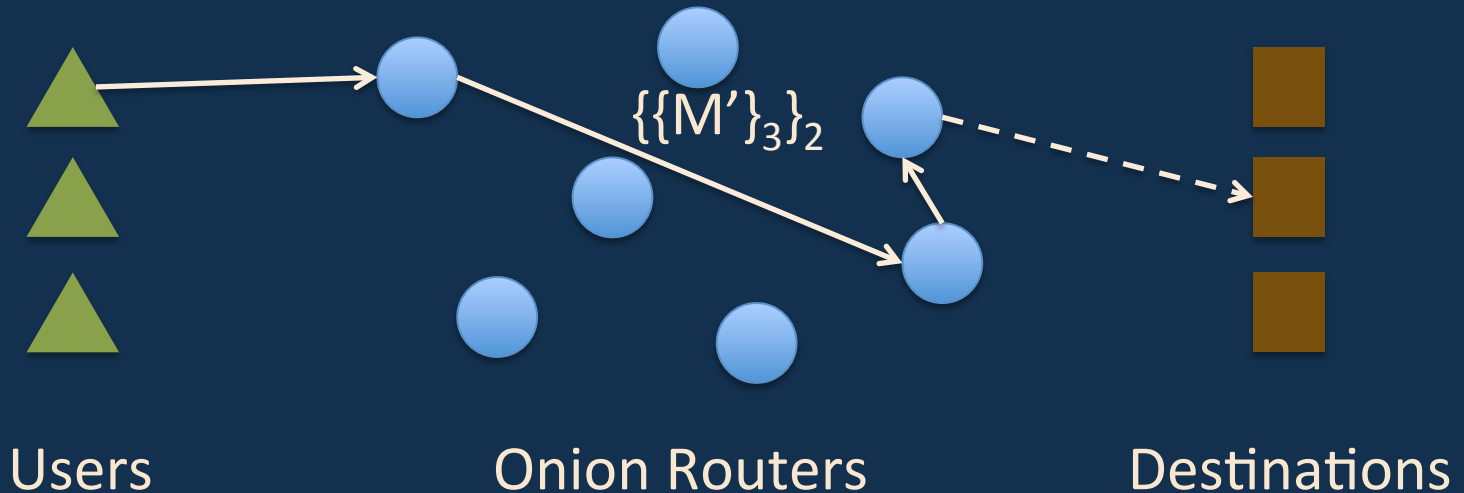
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.
4. The process runs in reverse for return data.

Onion Routing



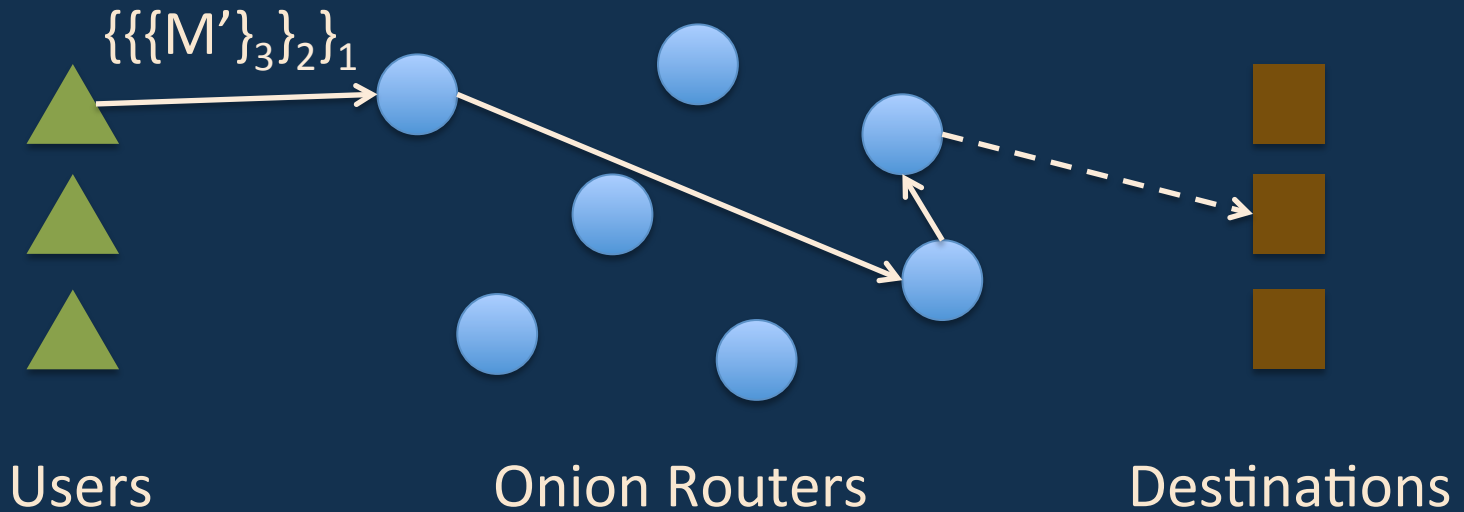
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.
4. The process runs in reverse for return data.

Onion Routing



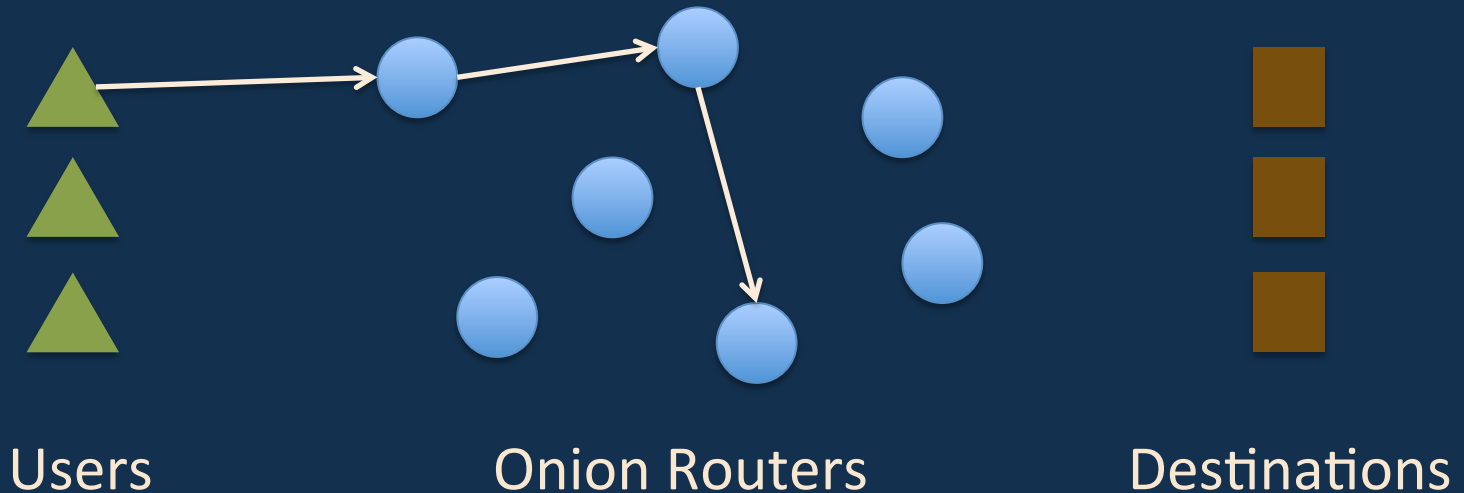
1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.
4. The process runs in reverse for return data.

Onion Routing



1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.
4. The process runs in reverse for return data.

Onion Routing



1. A user cryptographically constructs a circuit through the network.
2. A stream to the destination is opened.
3. Onion-encrypted data is sent and unwrapped along the circuit.
4. The process runs in reverse for return data.
5. The user changes the circuit periodically.

Problems

1. What is trust?

- Model adversary
 - Differs among users
- Model user knowledge
 - Include uncertainty

2. How to use trust?

- Blend user connections together
- Use trust explicitly in guard nodes
- Protect trust information

Model

Agents

- Users: U
- Routers: R
- Destinations: D
- Adversaries: $\{A_u\}_{u \in U}$

- Naïve users: $N \subset U$
- $A_{n_1} = A_{n_2}, n_1, n_2 \in N$
- $c^n(r) = c_N, n \in N$

Trust

- Probability of compromise: $c^u(r)$
- Trust: $\tau^u(r) = 1 - c^u(r)$
- Known whether source and destination links are observed

Anonymity Analysis

Expected anonymity	Downhill	Most trusted	Random	Lower bound
Many @ medium trust	0.0274	0.2519	0.1088	0.01
Many @ low trust	0.0550	0.1751	0.4763	0.001
Equal high/med/low	0.1021	0.1027	0.5000	0.1

Scenario 3: Trust is based on geographic region.

