# Anonymous Routing (With Emphasis on Tor)

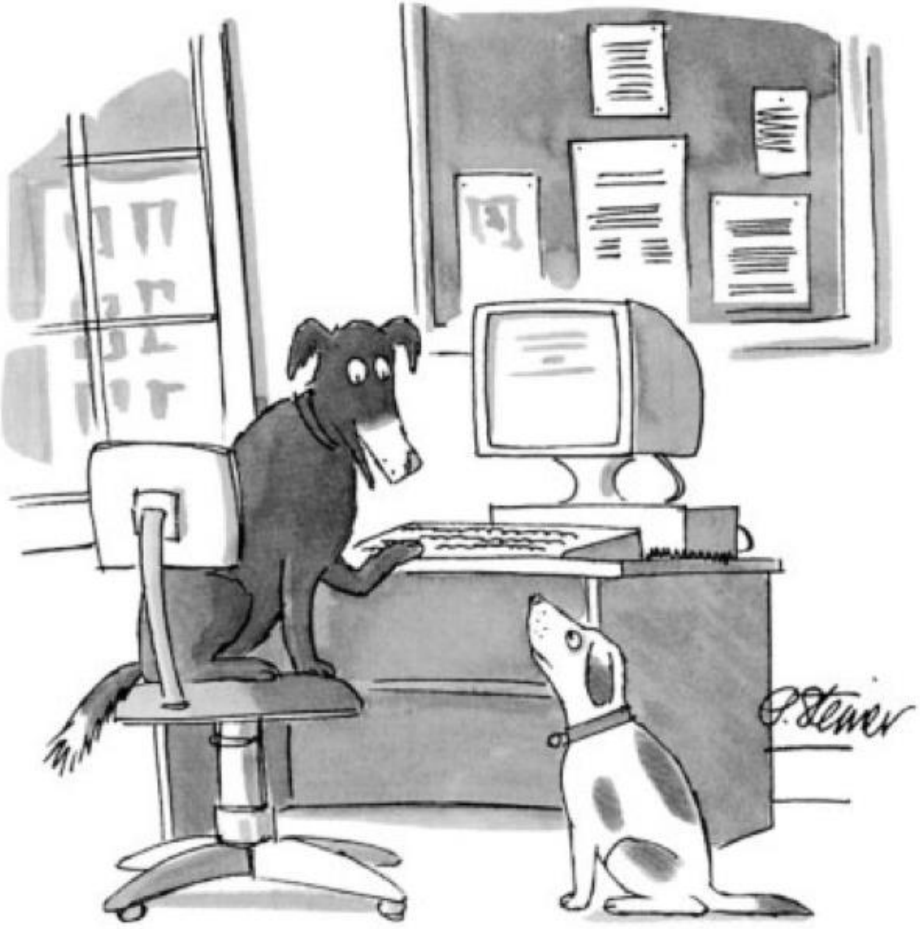
Erman Ayday

# 20th century perception of the Internet

*"On the Internet, nobody knows you're a dog."*

NY Times, July 5, 1993

# 21$^{th}$ century perception of the Internet



Credit:
Arvind
Narayanan

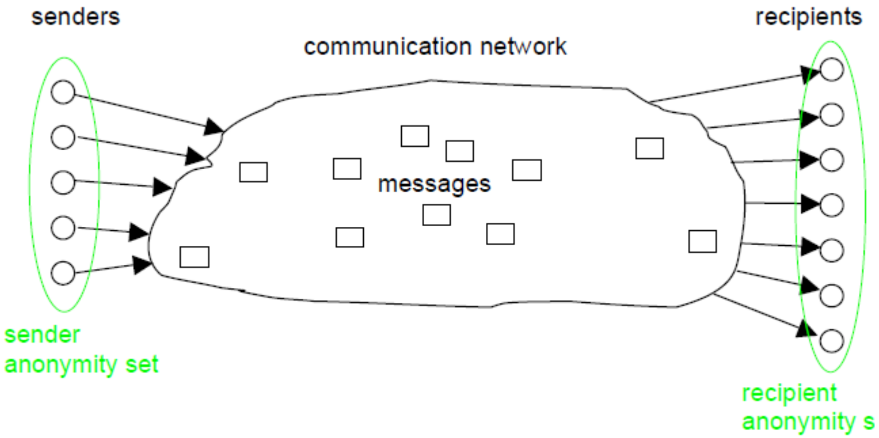
*It’s the Internet! Of course they know you’re a dog. They also know your favorite brand of pet food and the name of the cute poodle at the park that you have a crush on!*

# Reality

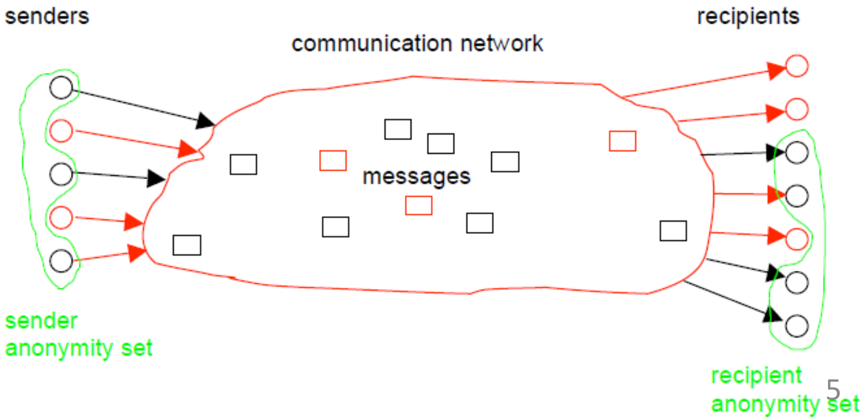
- Today' s communications infrastructure is capable of identifying and recording
  - Who we are
  - What we do
  - Where we go
  - What we say
  - What we buy
  - Who are our friends/family
- Anonymity on the Internet seems to be highly desirable

# Reminder - Anonymity

- Anonymity: state of being not identifiable within a set of subjects, the anonymity set
- Anonymity is stronger if:
  - larger anonymity set
  - even distribution of the sending and/or receiving



largest possible anonymity sets



largest possible anonymity sets w.r.t. attacker

# Anonymous Communications

- Infrastructure running on top of the existing Internet protocols
- Allow people to communicate without revealing their personal network IDs
  - E.g., IP addresses
- Provide a strong foundation for censorship resistance
  - Particularly well suited for people living under oppressive regimes

# Who Needs It?

- Regular citizens
  - Against a stalker or censor
- Law enforcement
  - Against an investigated suspect
    - “Why is alice.fbi.gov visiting my website?”
- Companies
  - Against a competitor
    - “Hey it’s Alice, give her the ‘Alice’ version”
- Governments
  - Against untrusted ISPs

# Traffic Analysis

- Ignores the content of messages
  - Payloads of the packets can be encrypted
- Tries to derive as much information as possible from only the payload and network traffic metadata
  - Source - destination
  - Packet arrival times
  - Message lengths

# Adversarial Models

- Capability
  - Passive: able to monitor and record the traffic on network links
  - Active: “Passive” + ability to inject/modify/delete traffic
- Visibility
  - Partial
  - Global
- Mobility (in case visibility is partial)
  - Static
  - Adaptive
- Participation
  - External
  - Internal (has compromised one or several clients / pieces of the network infrastructure)

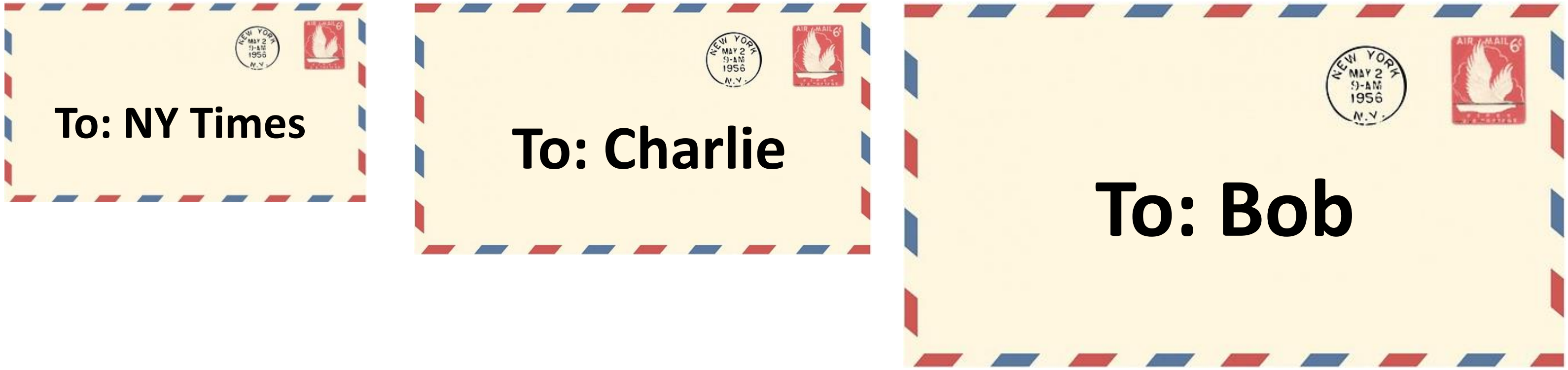
# Anonymity Systems

- High-latency anonymity systems
  - Mixes
  - Mix networks
- Low-latency anonymity systems
  - Anonymizer.com
  - Onion routing
  - Crowds
  - Tor (anonymity network)

# High-Latency Anonymity Systems

- Provide strong anonymity
- Only applicable for applications that can tolerate delays of several hours or more
- Example:

Alice anonymously sends a letter to NY Times

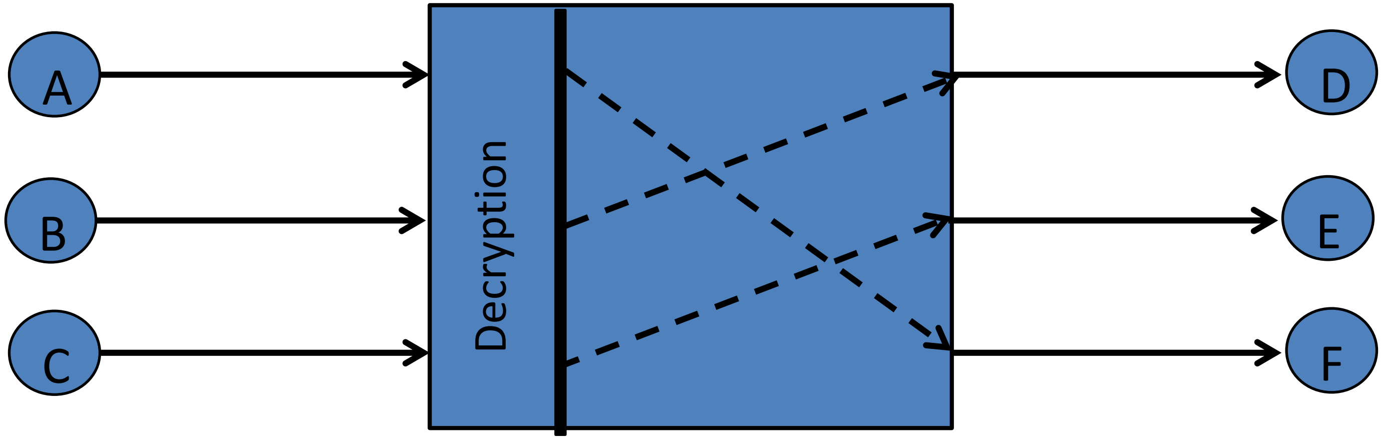


Bob only knows the next hop in the path and not the letter’s content or its intended destination

# Mixes

- Introduced by Chaum, 1981
- Idea:
  - Accepts encrypted messages as input,
  - Groups several messages together into a *batch*
  - Decrypts and forwards some or all of the messages
- Example:
  - Alice wants to anonymously send a message $M$ to a recipient Bob *via* a single mix $x$
  - Alice computes $E_x(R_x, M, A_B)$ and sends to the mix
    - $R_x$ is a string of random bits, and $A_B$ is Bob's address
    - Mix decrypts the input and forwards $M$ to Bob based on the mixing strategy
- An observer cannot link messages based on their arrival and departure times
  - Messages are delayed, batched, and reordered
- Sender must trust the mix

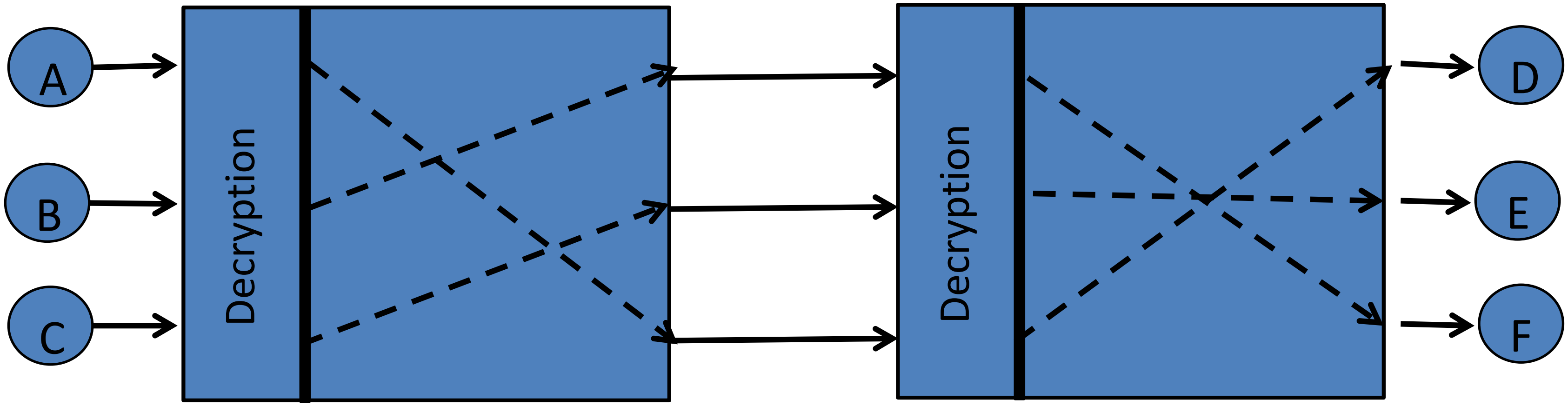
# Mix: Example



Input to the mix E(message, IP_Dest)

Risk of timing attacks → the mix should introduce random delays on the packet traversal; it can also generate dummy packets to further confuse the adversary

# Mix Cascading



Unless all mixes are compromised, no one can know who communicates with whom ➔ unlinkability

An identifier (pseudonym) can be used to identify a given path (in case multiple messages need to be sent)

# Mix Networks

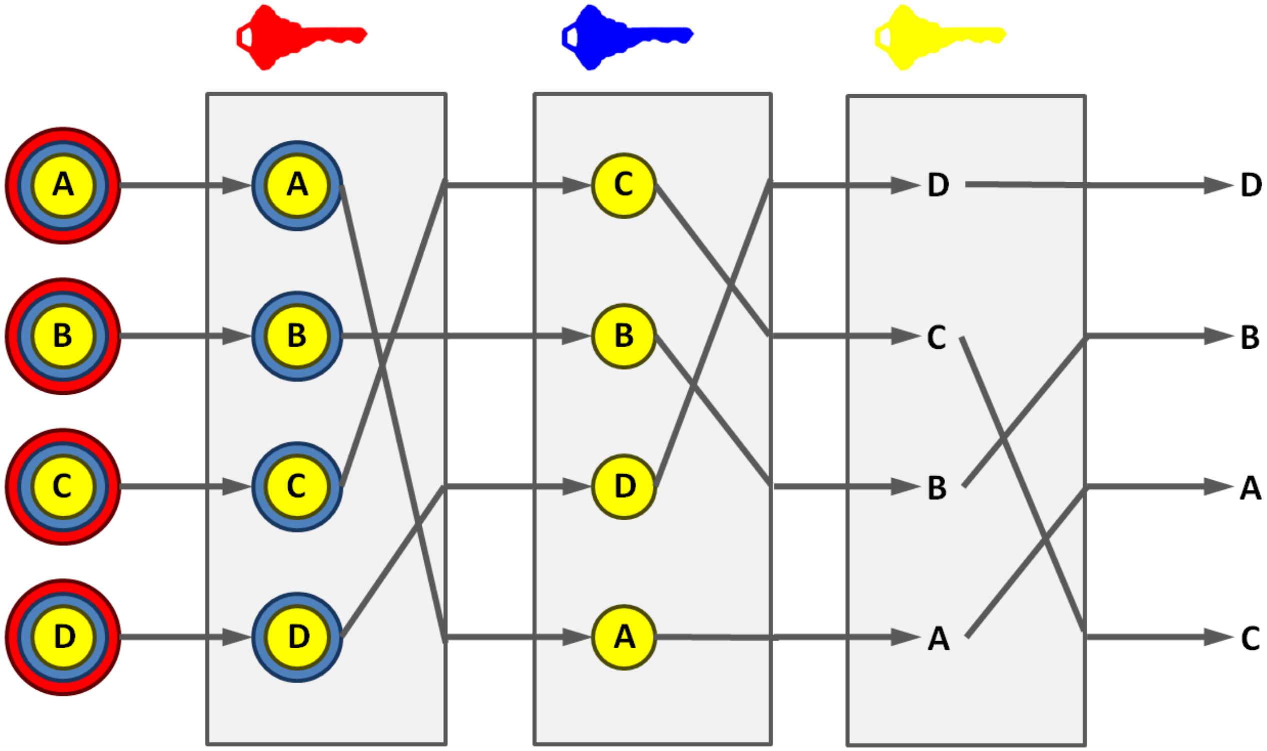


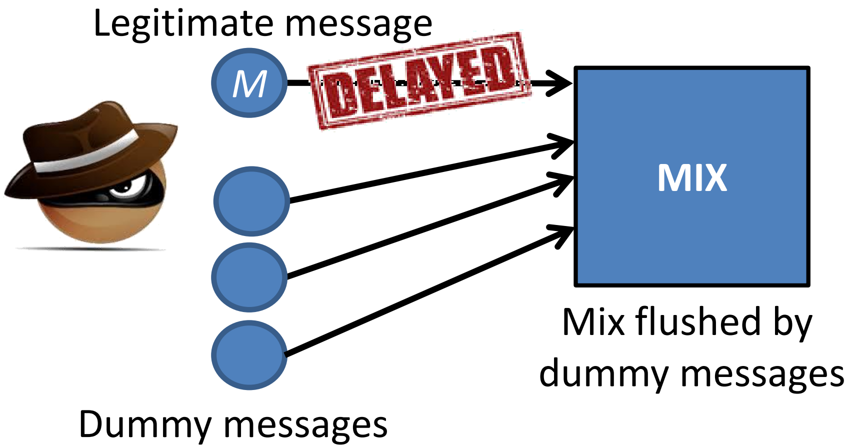
Figure: Wikipedia

- Sequence of mixes
- Each mix along a message' s path knows only the hop before and after itself
- An identifier (pseudonym) can be used to identify a given path (in case multiple messages need to be sent)

# Flushing Algorithms

- Determine which messages to forward to their next destination and when to do so
- Threshold mixes
  - Collects incoming encrypted messages until it receives n messages
  - Then decrypts all of them and forwards them to their next destination in  random order

# (n-1) attack

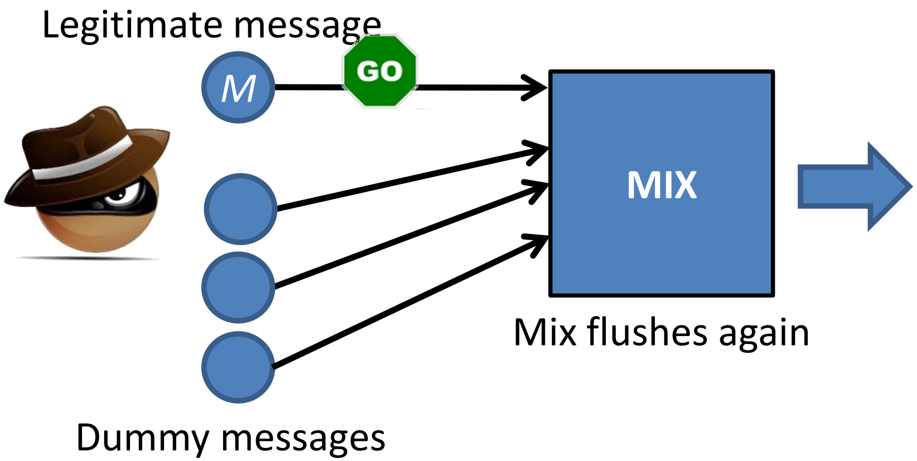


Legitimate message

Dummy messages

Mix flushed by dummy messages

Most mixes are subject to (n-1) attack



Legitimate message

Dummy messages

Mix flushes again

Only mix output that is not one of the dummy messages corresponds to *M*

# (n-1) attack

- Also called “flooding attack”
- Attacker delays a legitimate message *M* about to enter the mix
- Generates and sends his “dummy messages” into the mix until it flushes
- Then allows *M* into the mix and sends more of his dummy messages until the mix flushes again
- The only mix output that is not one of his dummy messages thus corresponds to input *M*

# Flushing Algorithms

- **Timed mixes**
  - Collects messages for a fixed length of time *t*
  - Vulnerable to an active attack similar to the (n−1) attack (called “trickle attack”)
- **Threshold and/or times mixes**
  - Collects messages until either it has received *n* messages *or* until *t* seconds have elapsed
  - Still vulnerable to (n-1) and trickle attacks
- **Pool mixes**
  - Selects a random subset of the collected messages to flush
  - Waits until it receives *n* messages in addition to the *Np* messages in the pool
  - Randomly selects *n* messages from the *n*+*Np* total messages in the mix to forward
  - Timed pool mixes operate similarly
  - *Dynamic* pool mixes send a fraction of the total messages in the pool
  - (n-1) attack becomes probabilistic

# Flushing Algorithms

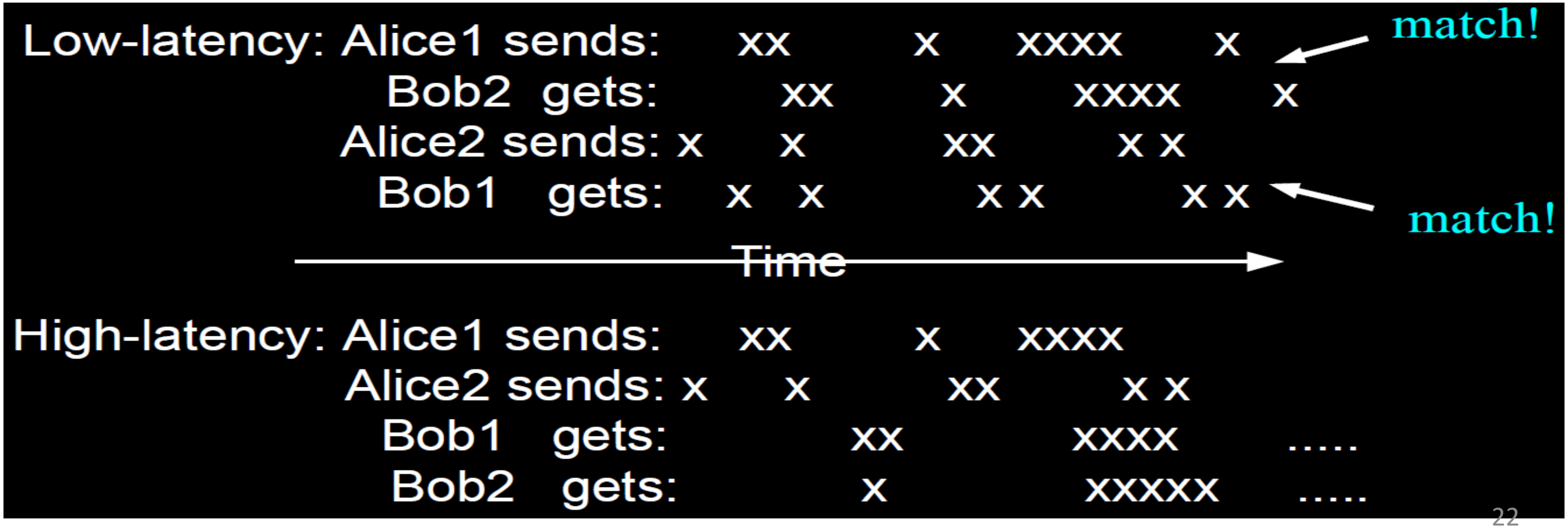
- **Stop-and-go mixes**
  - Instead of batching messages together, individually delay messages
  - Delay is based on exponential distribution
- **Binomial mixes**
  - At the end of a mix round lasting *t* seconds, a binomial mix flips a biased coin with a forwarding probability $p_f = P(n)$ for each message
  - $P(n)$ is the coin's bias when the mix contains *n* messages
- **RGB (Red Green Black) mixes**
  - Can detect the (n-1) attack
  - Genuine client messages received during a mix round are considered *black* messages
  - To estimate the number of legitimate messages received during a round, mixes send out "heartbeat" traffic, called *red* messages
    - Red traffic is "anonymously" addressed back to the mix itself
  - If the mix detects a statistically significant drop in its received red traffic, it can deduce that it is currently under attack
    - If there is an attack, red messages will delay a lot due to attacker's dummy messages
  - The mix generates additional dummy messages, called *green* traffic
    - To increase the anonymity of the legitimate, black messages output during each round

# Deployed High-Latency Anonymity Systems (all dead or dying)

- anon.penet.fi
  - Simple, single-server system
  - Shut down in 1996 after lawsuit from Church of Scientology
- Cypherpunk Remailers (or Type I)
  - Loosely based on Chaum's mix design
  - Multiple, distributed servers
- Mixmaster (or Type II)
  - Improved version of Cypherpunk Remailers
  - Uses a timed dynamic pool flushing algorithm
- Mixminion (or Type III)
  - Includes directory servers for clients to learn about remailers
  - http://mixminion.net

# Low-Latency Anonymity Systems

- Flushing algorithms introduce large delays
- Need to improve performance for real-time applications (web browsing,…)
- Susceptibility to certain traffic analysis attacks increases



Low-latency: Alice1 sends: xx x xxxx x

Bob2 gets: xx x xxxx x

Alice2 sends: x x xx x x

Bob1 gets: x x x x x x

match!

match!

Time

High-latency: Alice1 sends: xx x xxxx

Alice2 sends: x x xx x x

Bob1 gets: xx xxxx .....

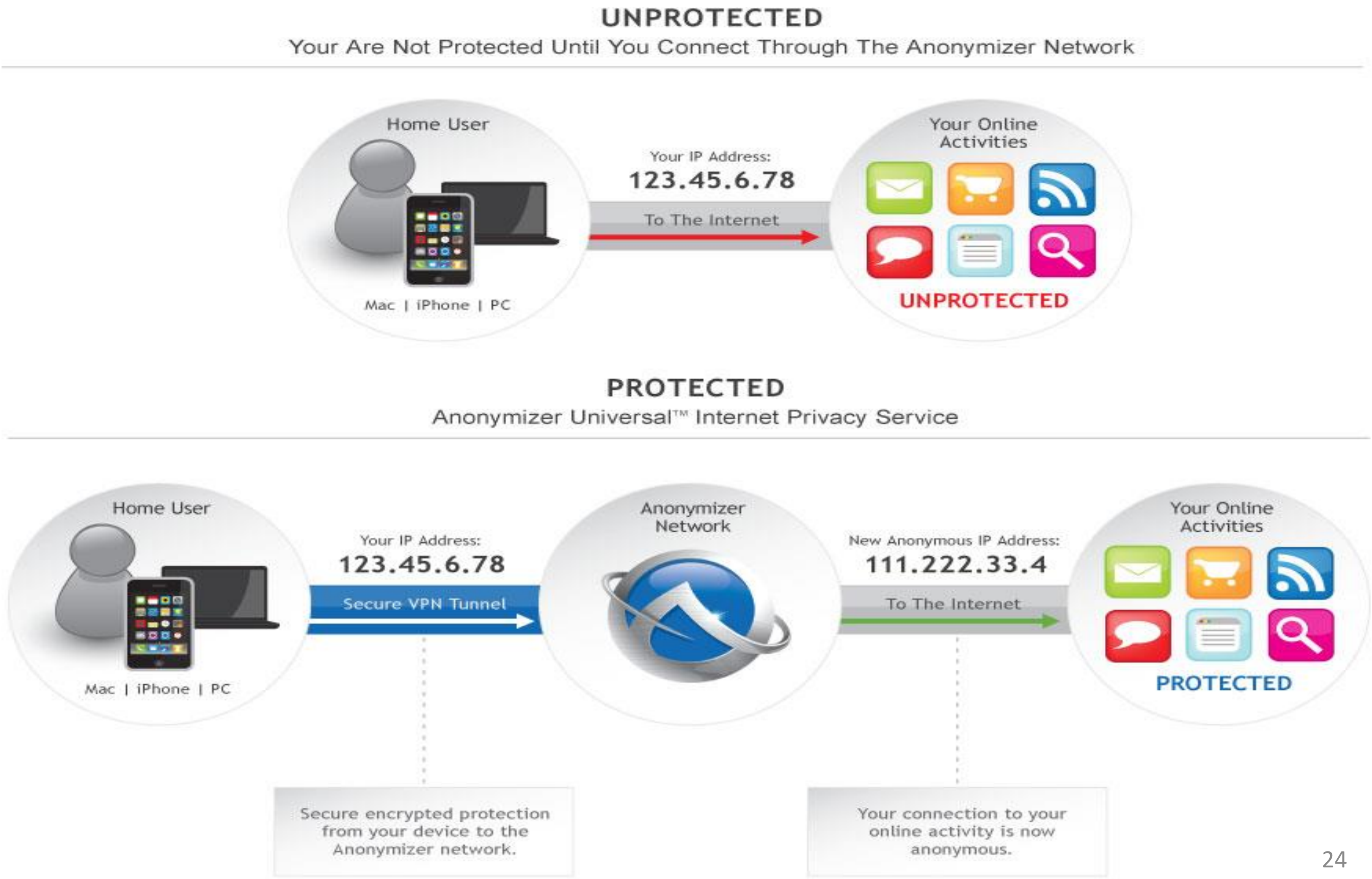
Bob2 gets: x xxxxx .....

Figure: Paul Syverson, U.S. Naval Research Laboratory

# anonymizer.com

- One of the first Web privacy companies (1995)
- Creates a VPN link between its servers and users' computers, creating a random IP address
- Based on the notion of *proxy*
- Forward all incoming traffic (e.g., a TCP connection) immediately without any packet reordering
- Requires clients to trust the company to not monitor or log their traffic

# anonymizer.com - Example



UNPROTECTED

Your Are Not Protected Until You Connect Through The Anonymizer Network

PROTECTED

Anonymizer Universal™ Internet Privacy Service

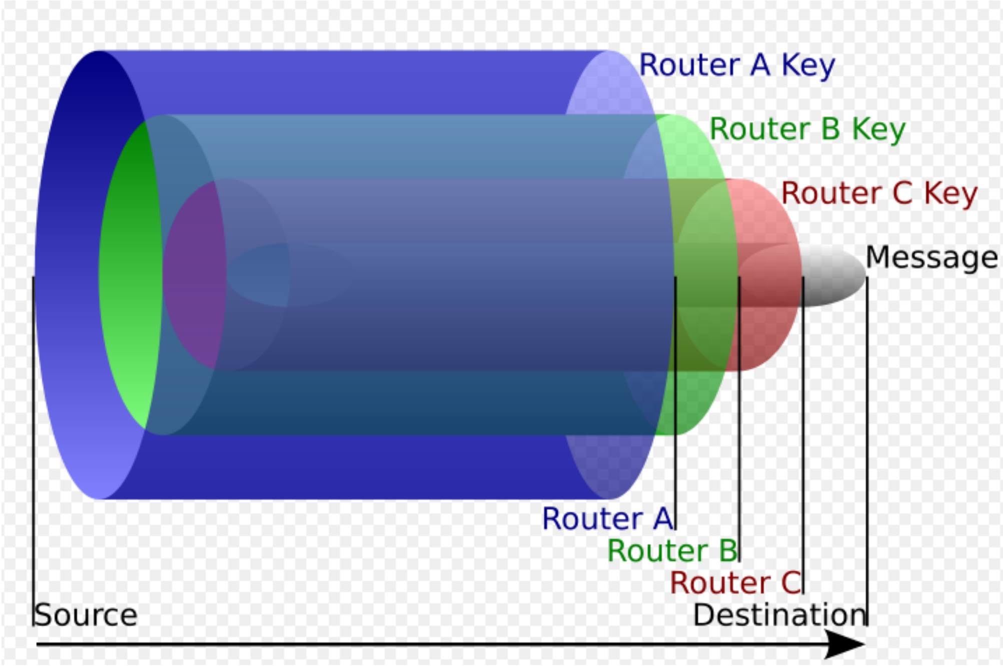
Secure encrypted protection from your device to the Anonymizer network.

Your connection to your online activity is now anonymous.

# Onion Encryption and Onion Routing

- Set of servers called *onion routers* that relay traffic for clients
- Route: Source → Relay A → Relay B → Relay C → Destination
- Uses public key cryptography to establish the encrypted circuit
- Uses faster **symmetric key cryptography** to transfer the actual data
- Before transmission, the Source performs onion encryption:
- $E_{Pu_A}(E_{Pu_B}(E_{PuK_C}(E_{PuK_{Dest}}(\text{message}))))$

- Node A "peels off" the first layer of encryption, Node B the next one, etc.



# Onion Routing – More Details

- Initiator choses the path $R_x \rightarrow R_y \rightarrow R_z$
- Initiator constructs the encrypted onion:

$$E_x\left(\begin{array}{c} t_x, f_x, KF_x, f_x^{-1}KB_x, R_y, \\ E_y\left(\begin{array}{c} t_y, f_y, KF_y, f_y^{-1}KB_y, R_z, \\ E_z(t_z, f_z, KF_z, f_z^{-1}KB_z, \emptyset) \end{array}\right) \end{array}\right)$$

$t_i$ indicates the expiration time
$KF_i$ is the forward key
$KB_i$ is the backward key
$f_i$ and $f_i^{-1}$ are forward and backward cryptographic functions (symmetric encryption schemes, in practice)

- $R_x$ removes the outermost layer of encryption using its private key, and learns the symmetric keys $KF_x$ and $KB_x$
- Once the circuit is constructed, the initiator can relay its application traffic over the circuit using the symmetric keys

# Crowds

- Designed for anonymous Web browsing
- Users in the system are represented by processes called *jondos* (a la "John Doe")
- When a user makes a Web request, his jondo randomly picks another jondo from the crowd and forwards the request
- jondo flips a coin biased with a forwarding probability $p_f >$ ½
- Depending on the outcome of the coin flip:
  - the jondo either randomly selects another member of the crowd to which the request will be forwarded, or
  - it forwards the request to the intended Web server
- Reply is relayed via the reverse of the established path
- When a jondo receives a request it does not know if that jondo was the original requester or if it simply forwarded the request for another jondo

# Tor (The Onion Router)

- Worldwide anonymous network
- Used by journalists, whistleblowers, bloggers, law enforcement officers,…
- Main protection is against **traffic analysis** (a threat not solved by simply encrypting the messages)
- **No individual relay** ever knows the complete path that a data packet has taken
- Only the first node (Router A) knows the IP address of the sender
- A longer route provides a higher guarantee of anonymity (in practice, Tor routing uses 3 relays)

# Tor (Anonymity Network)

TorProject.org

- Designed, implemented, and deployed as project of the U.S. Naval Research Laboratory
  - – R. Dingledine, N. Mathewson, and P. Syverson. Usenix Security 2004
- Distributes transactions over several places on the Internet
  - – Packets take a random pathway through several relays
  - – No observer at any single point can tell where the data came from or where it's going

# Tor - Overview

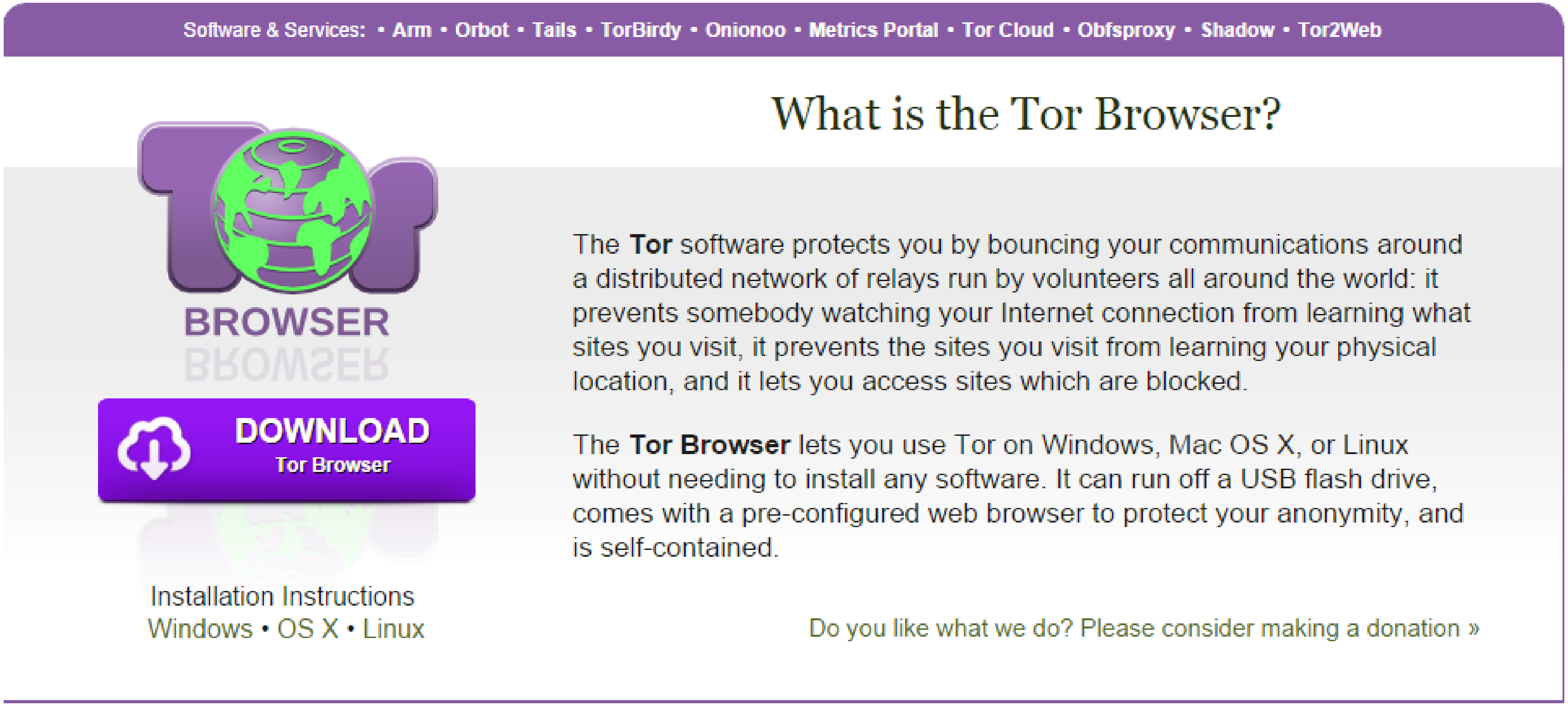
- Goal: to prevent attackers from linking communication partners, or from linking multiple communications to or from a single user
- Adversary's goal: try to link an initiator Alice with her communication partners, or try to build a profile of Alice's behavior
- Adversary model:
  - Can observe some fraction of network traffic
  - Can generate, modify, delete, or delay traffic
  - Can operate onion routers of his own
  - Can compromise some fraction of the onion routers
- Clients choose a path through the network and build a circuit
- Tor uses an incremental or *telescoping* path-building design
  - Initiator negotiates session keys with each successive hop in the circuit
  - Session keys are discarded after usage ➔ Tor users enjoy perfect forward secrecy
  - Each node in the path knows its predecessor and successor, but no other nodes in the circuit

# Tor – Use Cases

- Individuals use Tor to prevent websites from tracking them
  - E-commerce site may use price discrimination based on your country or institution of origin
- Individuals use Tor when specific services are blocked by their local Internet providers
- Journalists use Tor to communicate more safely with whistleblowers
  - Edward Snowden used Tor to send information about PRISM to the Washington Post and The Guardian
- Activist groups recommend Tor as a mechanism for maintaining civil liberties online
- Law enforcement uses Tor for visiting websites without leaving government IP addresses in their web logs

# Tor – How Can you Use It?

- Tor browser: withholds some information about your computer's configuration while browsing the Web



Software & Services: • Arm • Orbot • Tails • TorBirdy • Onionoo • Metrics Portal • Tor Cloud • Obfsproxy • Shadow • Tor2Web

## What is the Tor Browser?

The **Tor** software protects you by bouncing your communications around a distributed network of relays run by volunteers all around the world: it prevents somebody watching your Internet connection from learning what sites you visit, it prevents the sites you visit from learning your physical location, and it lets you access sites which are blocked.

The **Tor Browser** lets you use Tor on Windows, Mac OS X, or Linux without needing to install any software. It can run off a USB flash drive, comes with a pre-configured web browser to protect your anonymity, and is self-contained.

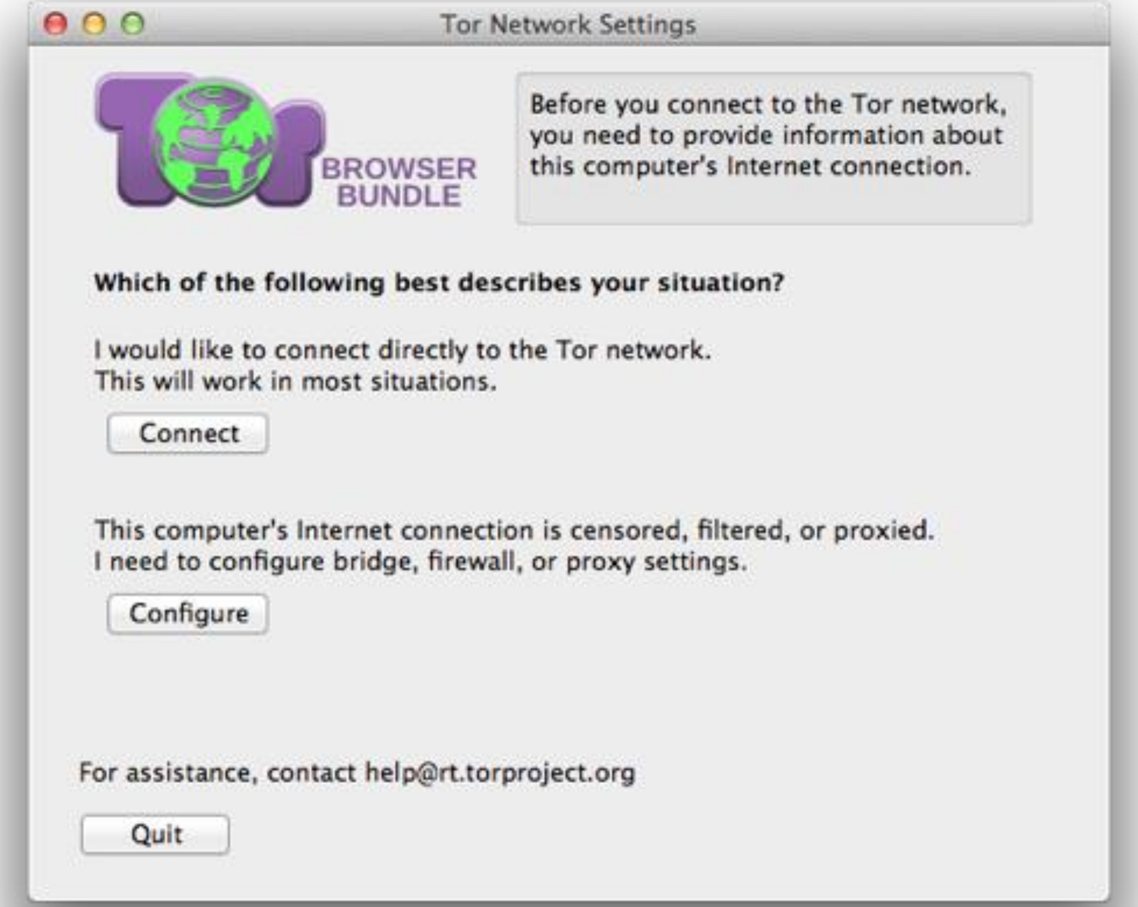
Installation Instructions
Windows • OS X • Linux

Do you like what we do? Please consider making a donation »

https://www.torproject.org/projects/torbrowser.html

# Getting in Touch with Tor

Download the Tor browser and then:



Before you connect to the Tor network, you need to provide information about this computer's Internet connection.

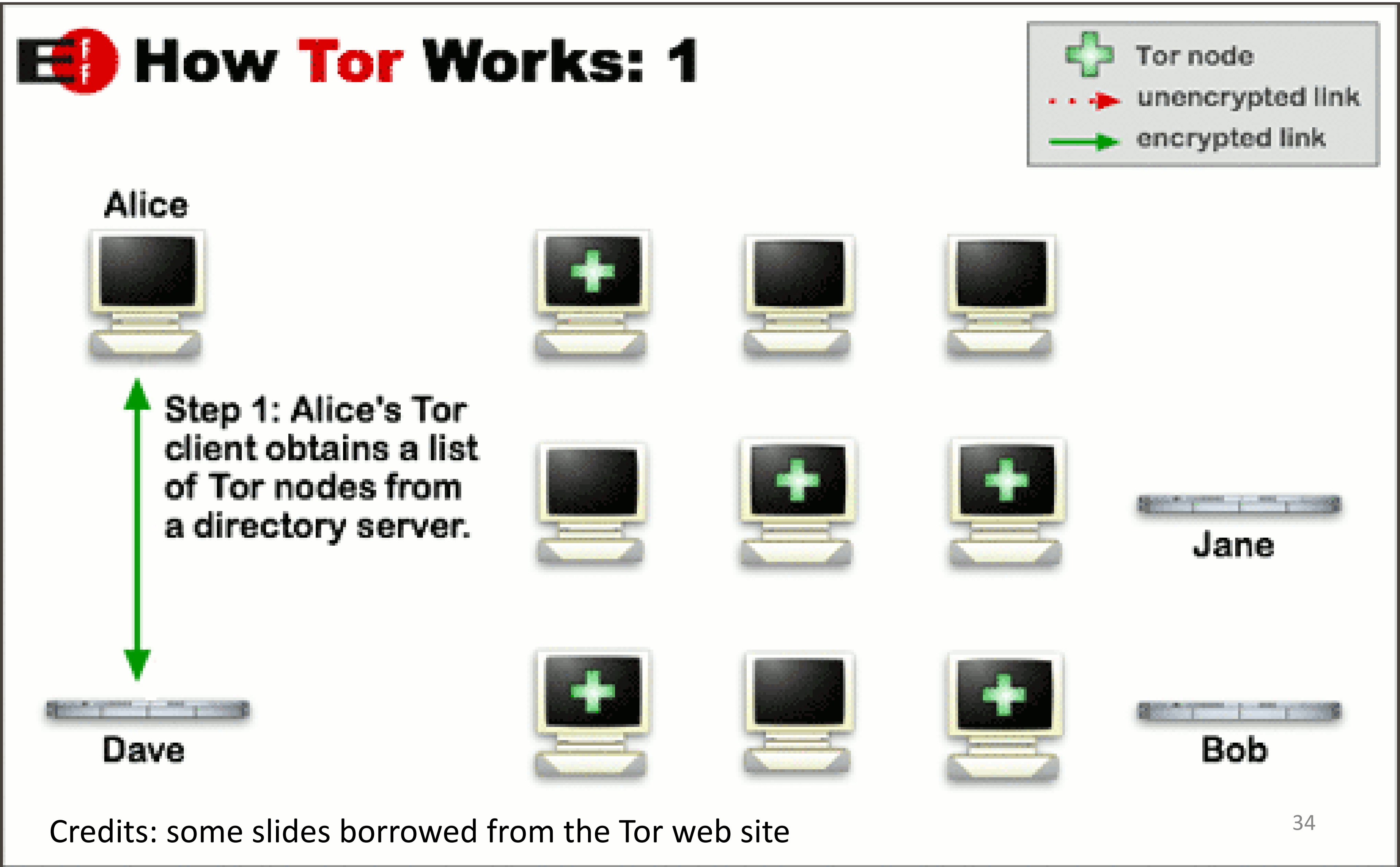
**Which of the following best describes your situation?**

I would like to connect directly to the Tor network.
This will work in most situations.

This computer's Internet connection is censored, filtered, or proxied.
I need to configure bridge, firewall, or proxy settings.

For assistance, contact help@rt.torproject.org

# Tor – Circuit Establishment



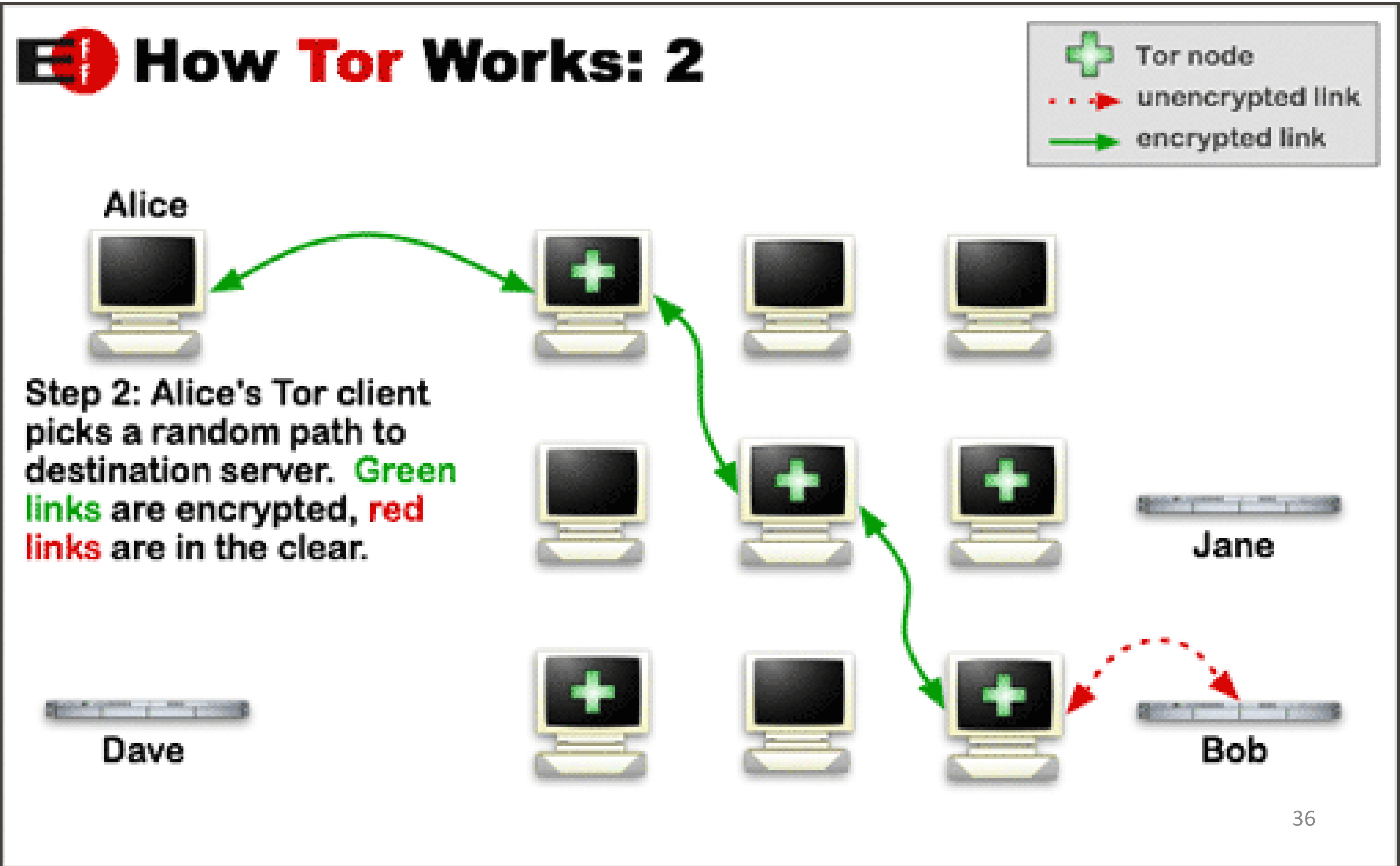
How Tor Works: 1
Tor node
unencrypted link
encrypted link
Alice
Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Jane
Dave
Bob

Credits: some slides borrowed from the Tor web site

# Tor – Circuit Establishment

- User's software/client incrementally builds a circuit of encrypted connections through relays (onion routers) on the network
  - Each user runs local software called an *onion proxy* to fetch directories and circuits across the network
  - The circuit is extended one hop at a time
  - Entry node -> relay -> exit node
- Each relay along the way knows only the previous and the next relays
- The client negotiates a separate set of encryption keys for each hop along the circuit
  - Each node in the path uses DH key negotiation

# Tor - Communication

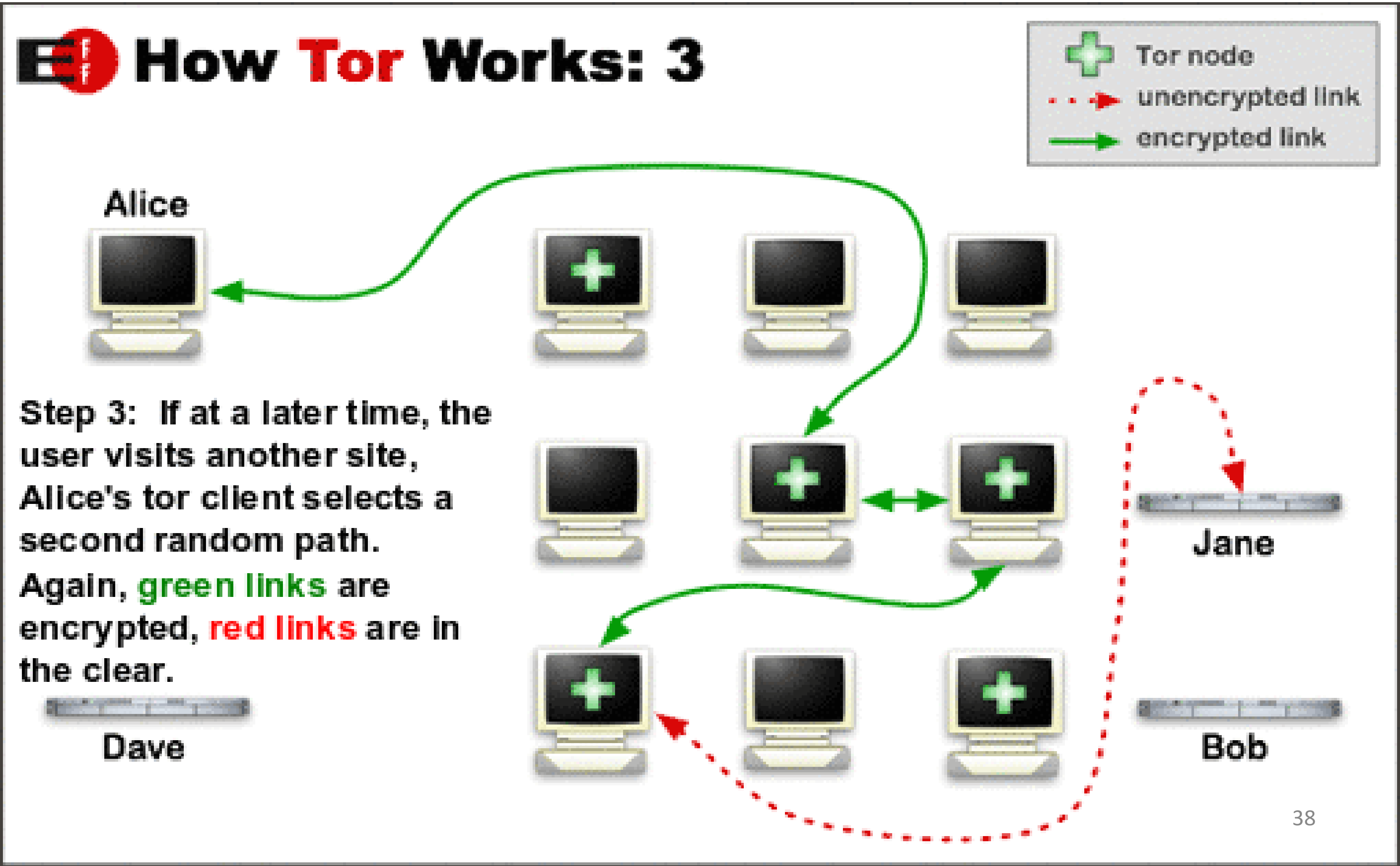


Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.

# Tor - Communication

- Onion proxies accept TCP streams and multiplex them across the circuits
- The onion router on the other side of the circuit connects to the requested destinations and relays data
- Neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination
- Uses the same circuit for connections that happen within the same ten minutes or so
  - Later requests are given a new circuit

# Tor - New Connection



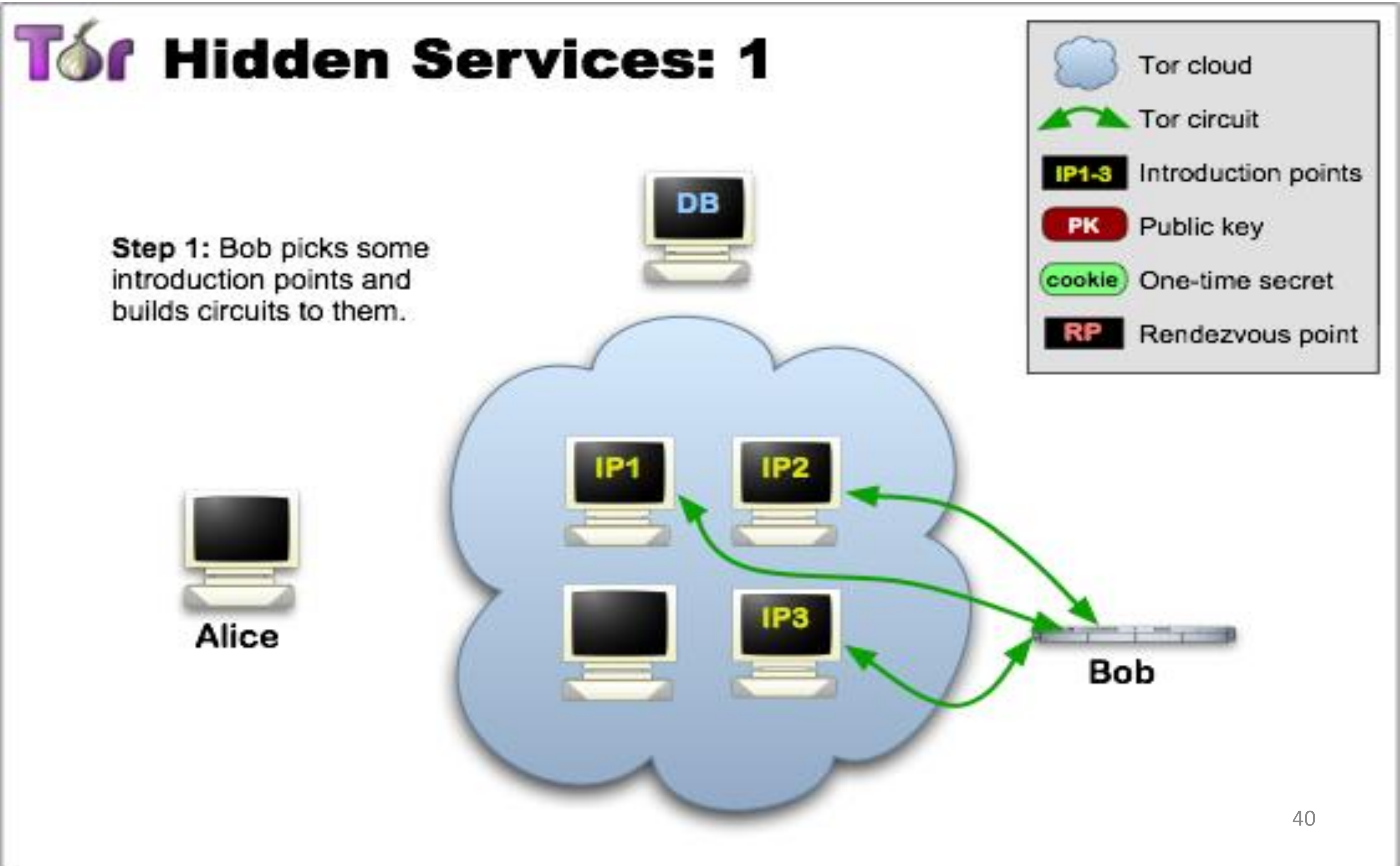
Step 3: If at a later time, the user visits another site, Alice's tor client selects a second random path.

Again, green links are encrypted, red links are in the clear.

# Tor – Hidden Services

- Sometimes also called "Location-Hidden Services"
- Let users publish websites and other services without revealing the location (IP address) of the site
  - Users can set up a website where people publish material without worrying about censorship
- Hidden services advertise their existence via "introduction points"
  - Know hidden service's public key (ID) but not its IP (location)
- Using Tor "rendezvous points", other Tor users can connect to the hidden services, each without knowing the other's network identity

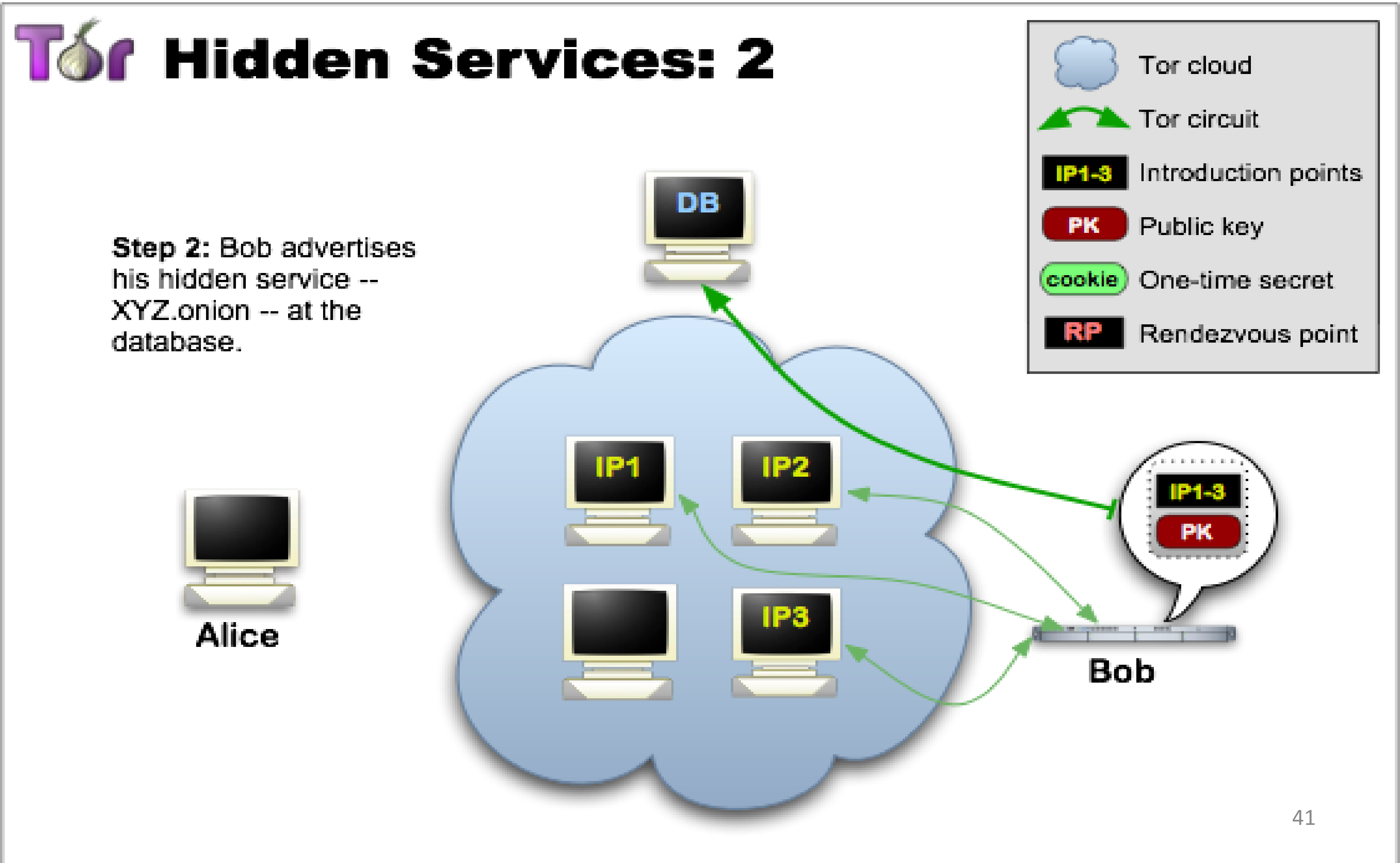
# Tor – Hidden Services



Tor Hidden Services: 1

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

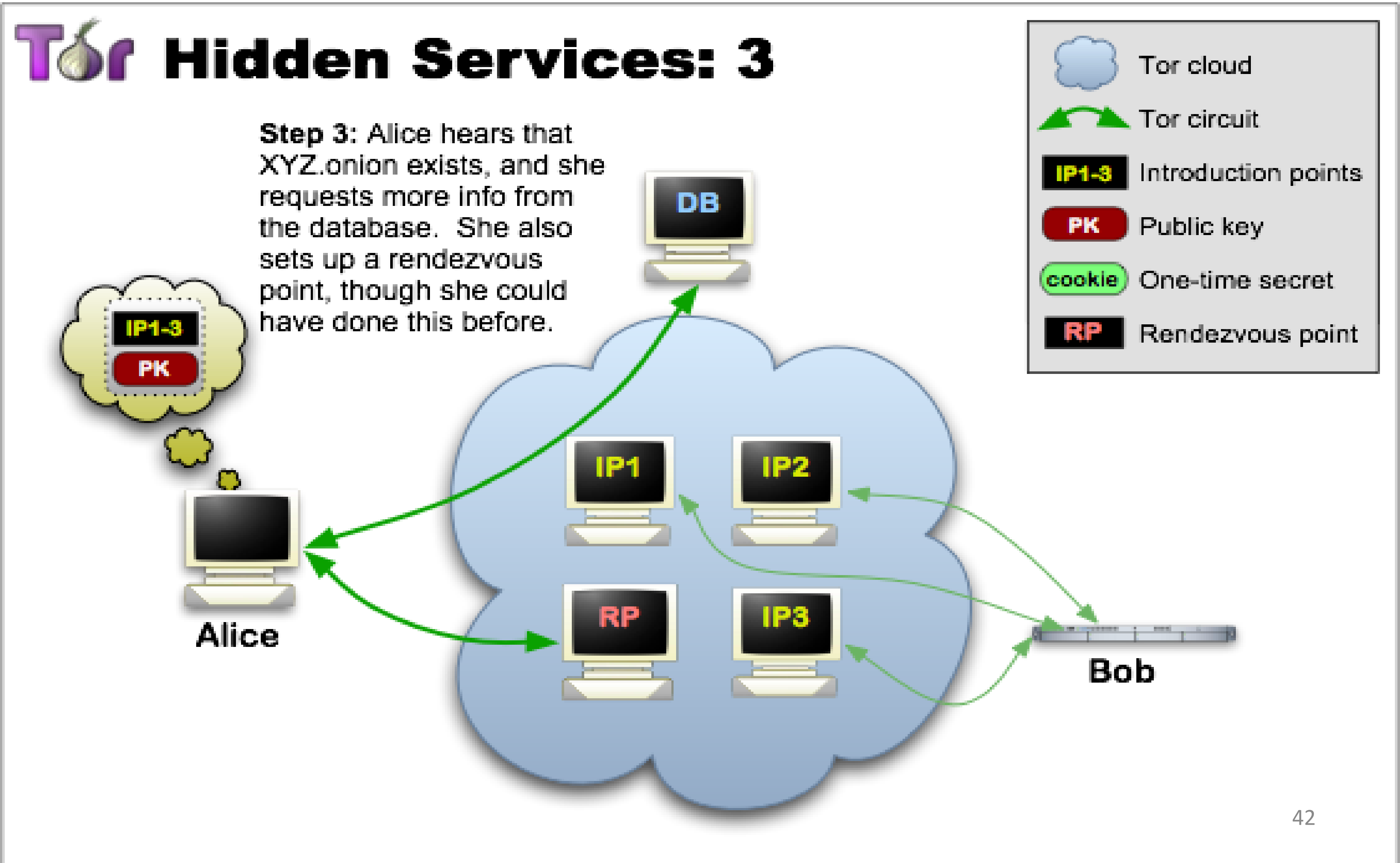
# Tor – Hidden Services



Tor Hidden Services: 2

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

# Tor – Hidden Services

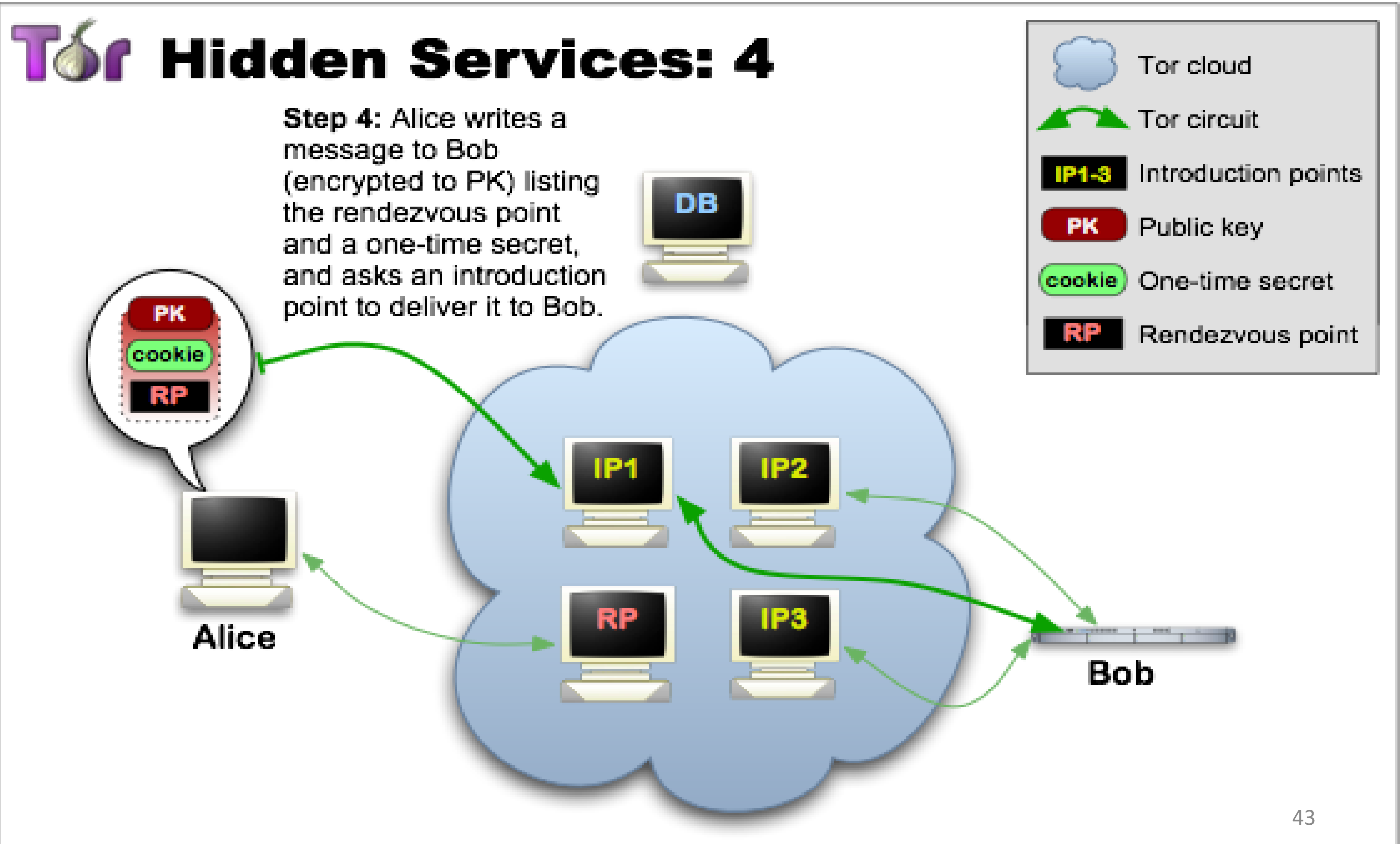


Tor Hidden Services: 3

**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 |

# Tor – Hidden Services

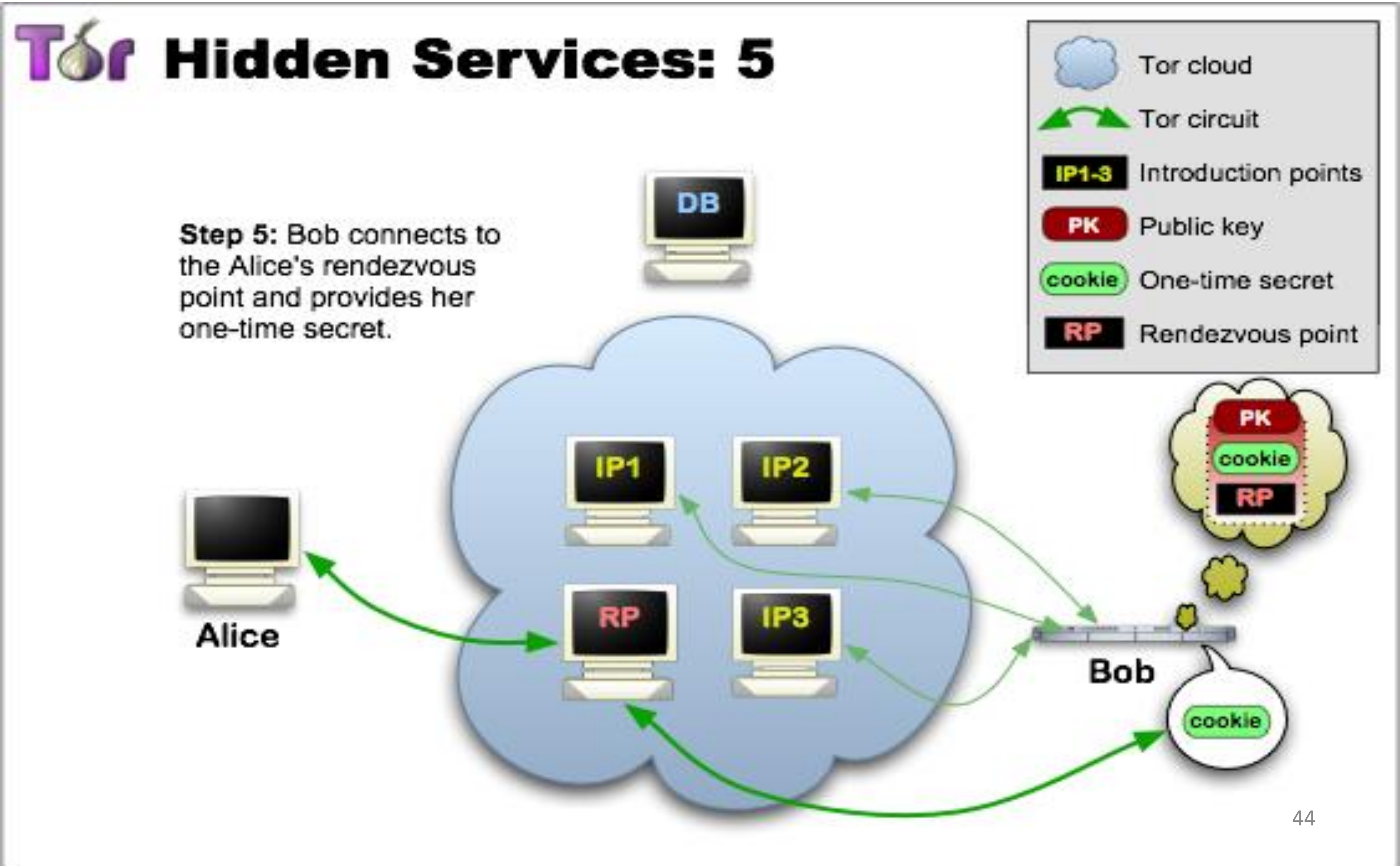


## Hidden Services: 4

**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.

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

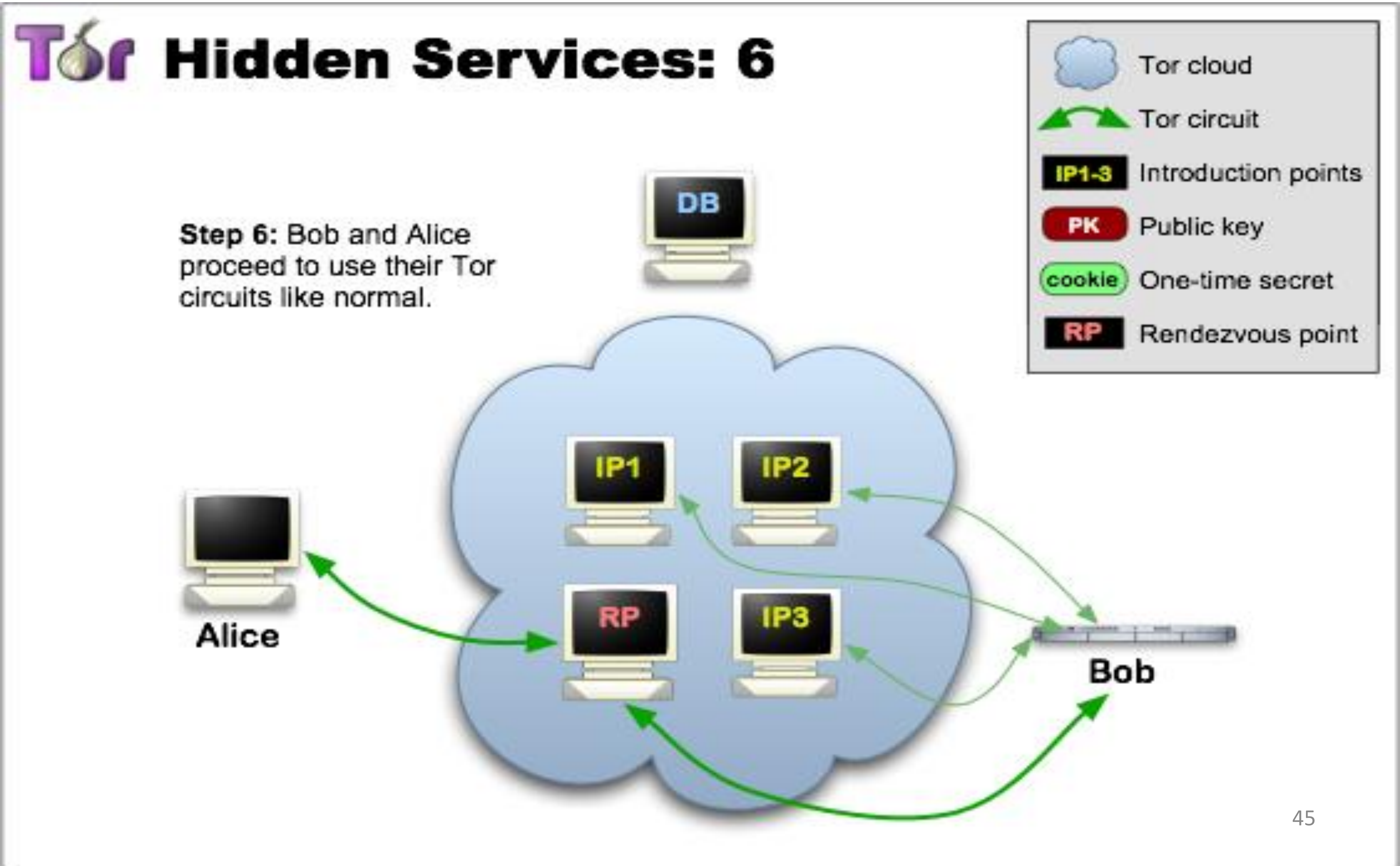
# Tor – Hidden Services



Tor Hidden Services: 5

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

# Tor – Hidden Services



Tor Hidden Services: 6

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

# Hidden Services - Discussion

- It is important that the hidden service sticks to the same set of *entry guards* when creating new circuits
  - Tor client selects a few relays at random to use as entry points, and uses only those for his first hop
    - Suppose the attacker controls, or can observe, *C* relays
    - Suppose there are *N* relays total
    - If you select new entry and exit relays each time you use the network, the attacker will be able to correlate all traffic you send with probability $(C/N)^2$
- Introduction circuit is not used for actual communication because no single relay should appear to be responsible for a given hidden service
  - Rendezvous point never learns about the hidden service's identity

# Tor - Bridges

- Bridges are Tor relays that aren't listed in the main Tor directory
- Even if your ISP is filtering connections to all the known Tor relays, they probably won't be able to block all the bridges
  - If you suspect your access to the Tor is being blocked, you may want to use the bridge feature
- To use a bridge:
  - you'll need to locate one
  - you'll need to configure Tor with whatever bridge address you intend to use

# Tor – Obfuscated Bridges

- Censors may find ways to block Tor even when clients are using bridges
  - E.g., installing boxes in ISPs that peek at network traffic and detect Tor
- Obfuscated bridges use special plugins called *pluggable transports* which obfuscate the traffic flow of Tor
  - E.g., Tor traffic flow looks like a Skype video
- Detection becomes harder
- Can be found in the BridgeDB

# Tor – Finding Bridges

- Visit the BridgeDB:

  https://bridges.torproject.org/

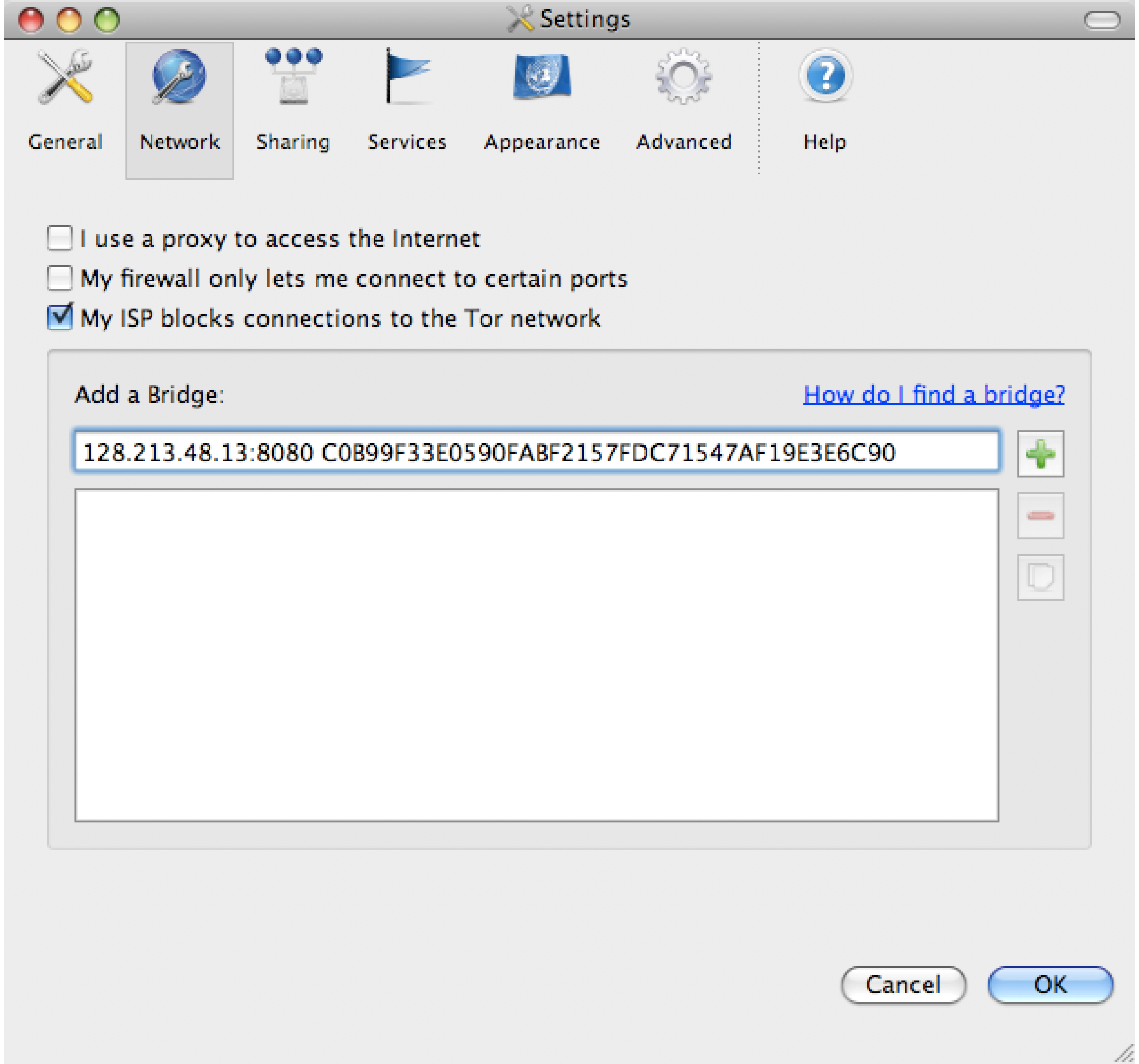
- Send e-mail to

  bridges@bridges.torproject.org

  - With the line “get bridges” in the body
  - From a gmail account to prevent attacks

- What you will receive
  - bridge 141.201.27.48:443 4352e58420e68f5e40bf7c74faddccd9d1349413
    - IP address: '141.201.27.48’
      Port: '443’
      Fingerprint (optional): '4352e58420e68f5e40bf7c74faddccd9d1349413 ‘
    - Fingerprint: identity of a bridge given by the bridge authority
  - bridge **obfs2** 141.201.27.48:420 4352e58420e68f5e40bf7c74faddccd9d1349413

# Tor - Adding a Bridge



□ I use a proxy to access the Internet

□ My firewall only lets me connect to certain ports

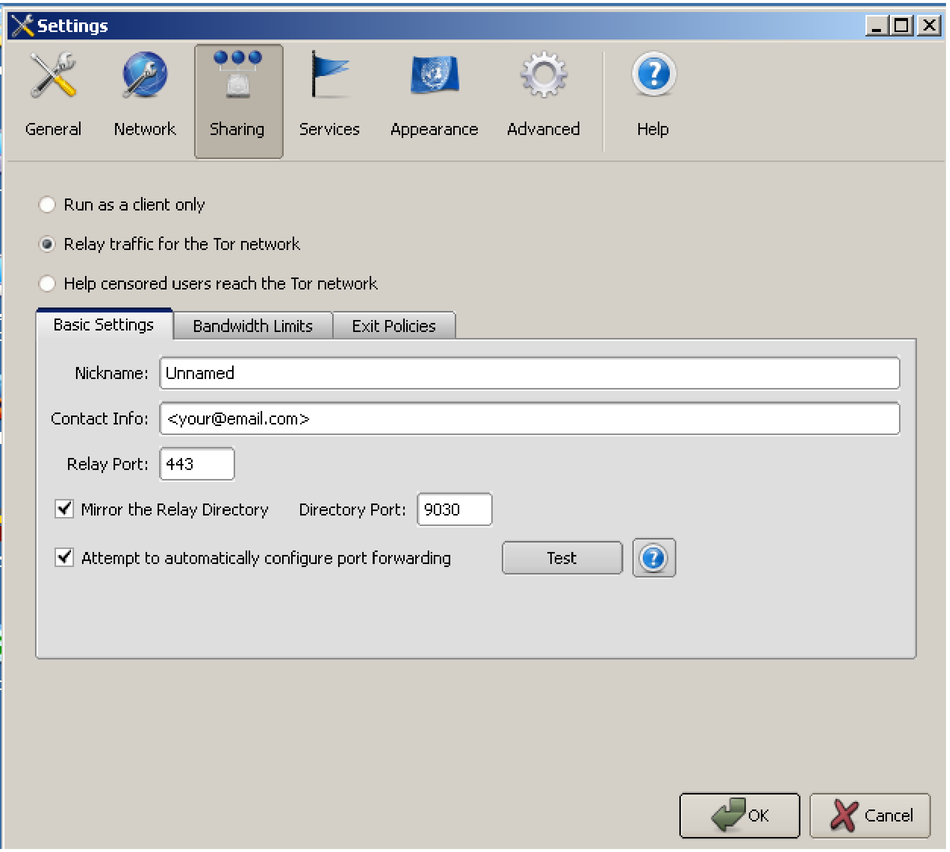
☑ My ISP blocks connections to the Tor network

Add a Bridge:

128.213.48.13:8080 C0B99F33E0590FABF2157FDC71547AF19E3E6C90

# Tor – How Can You Contribute?

- Running a Tor relay



○ Run as a client only

● Relay traffic for the Tor network

○ Help censored users reach the Tor network

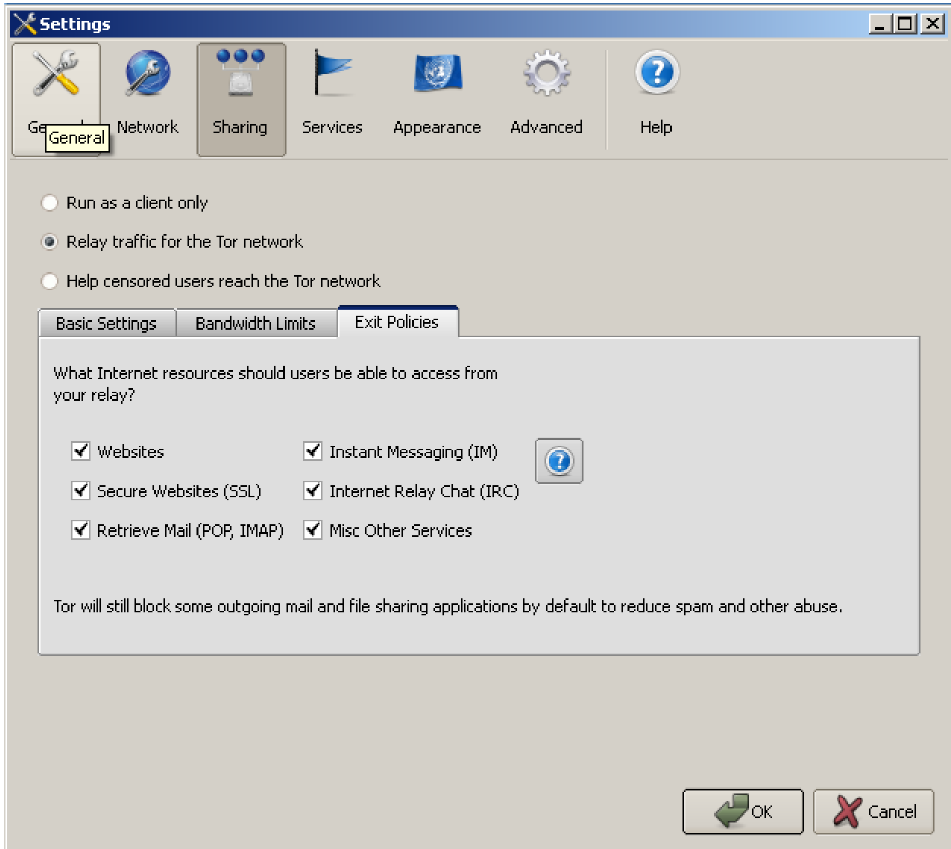
Nickname: Unnamed

Contact Info: <your@email.com>

Relay Port: 443

☑ Mirror the Relay Directory    Directory Port: 9030

☑ Attempt to automatically configure port forwarding



○ Run as a client only

● Relay traffic for the Tor network

○ Help censored users reach the Tor network

What Internet resources should users be able to access from your relay?

☑ Websites

☑ Secure Websites (SSL)

☑ Retrieve Mail (POP, IMAP)

☑ Instant Messaging (IM)

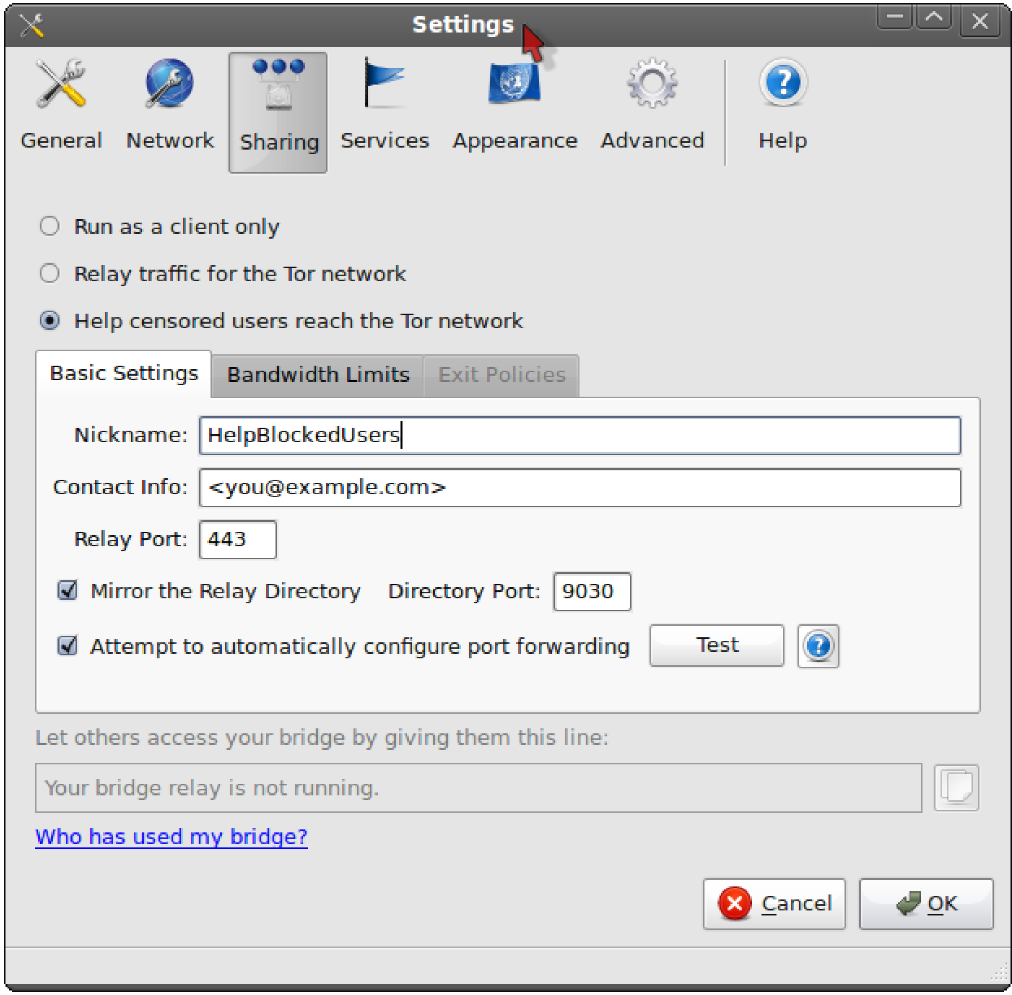
☑ Internet Relay Chat (IRC)

☑ Misc Other Services

Tor will still block some outgoing mail and file sharing applications by default to reduce spam and other abuse.

# Tor – How Can You Contribute?

- Running a Tor bridge:
  - Your bridge relay will automatically publish its address to the bridge authority
  - You can also tell a user about your bridge directly



○ Run as a client only

○ Relay traffic for the Tor network

● Help censored users reach the Tor network

Nickname: HelpBlockedUsers

Contact Info: <you@example.com>

Relay Port: 443

☑ Mirror the Relay Directory Directory Port: 9030

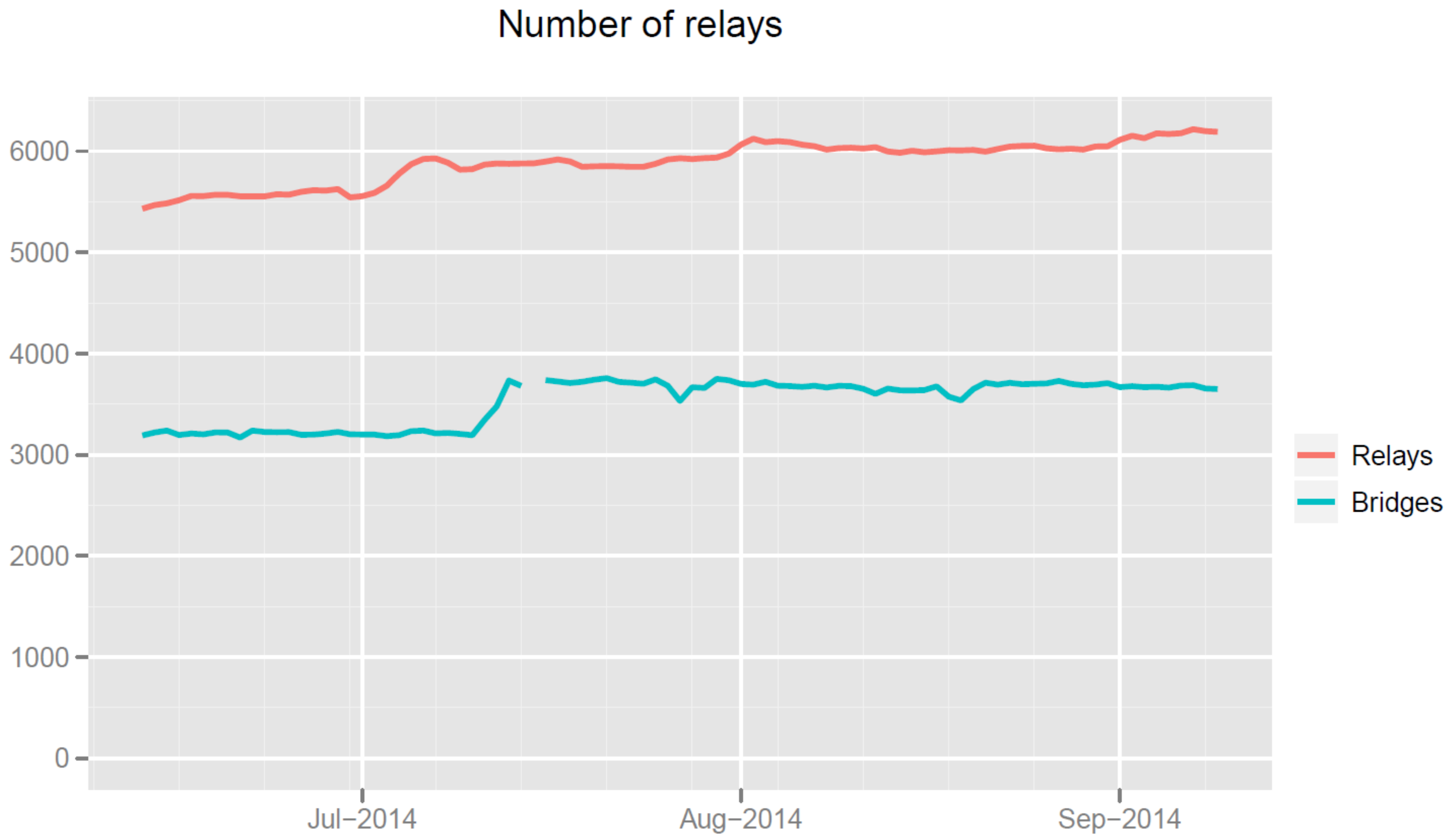
☑ Attempt to automatically configure port forwarding

Let others access your bridge by giving them this line:

Your bridge relay is not running.

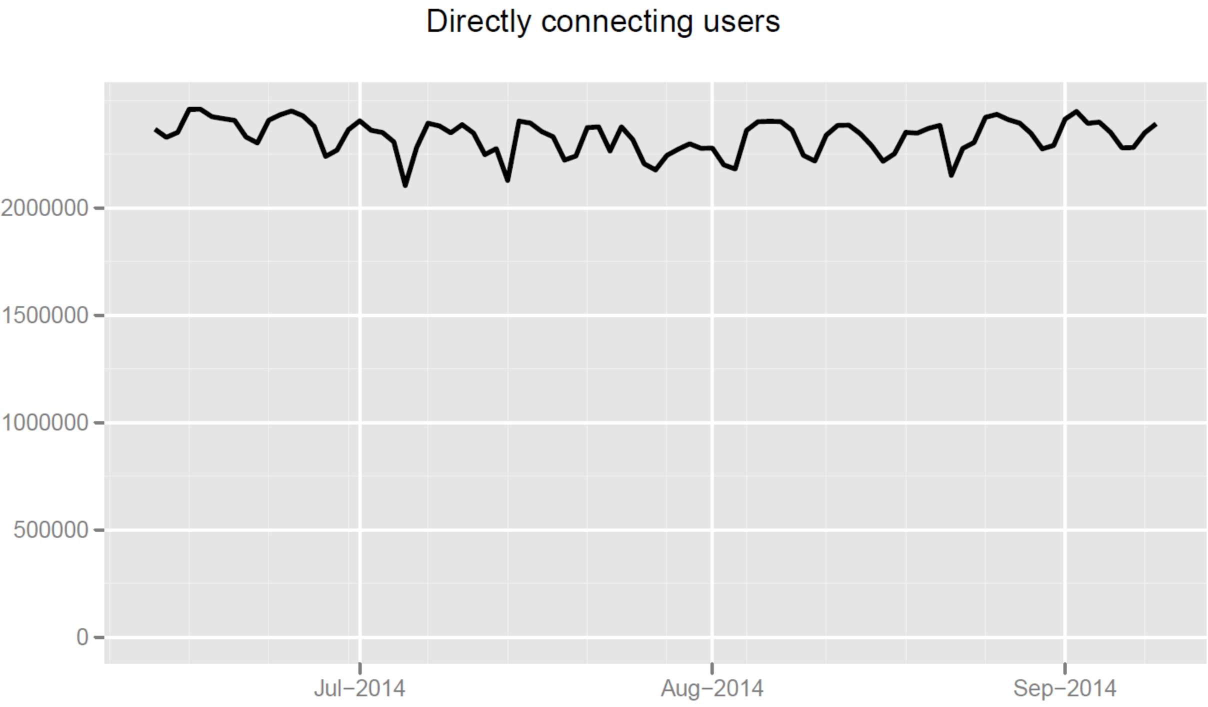
Who has used my bridge?

# Tor – Number of Relays



The Tor Project − https://metrics.torproject.org/

# Tor – Number of Users

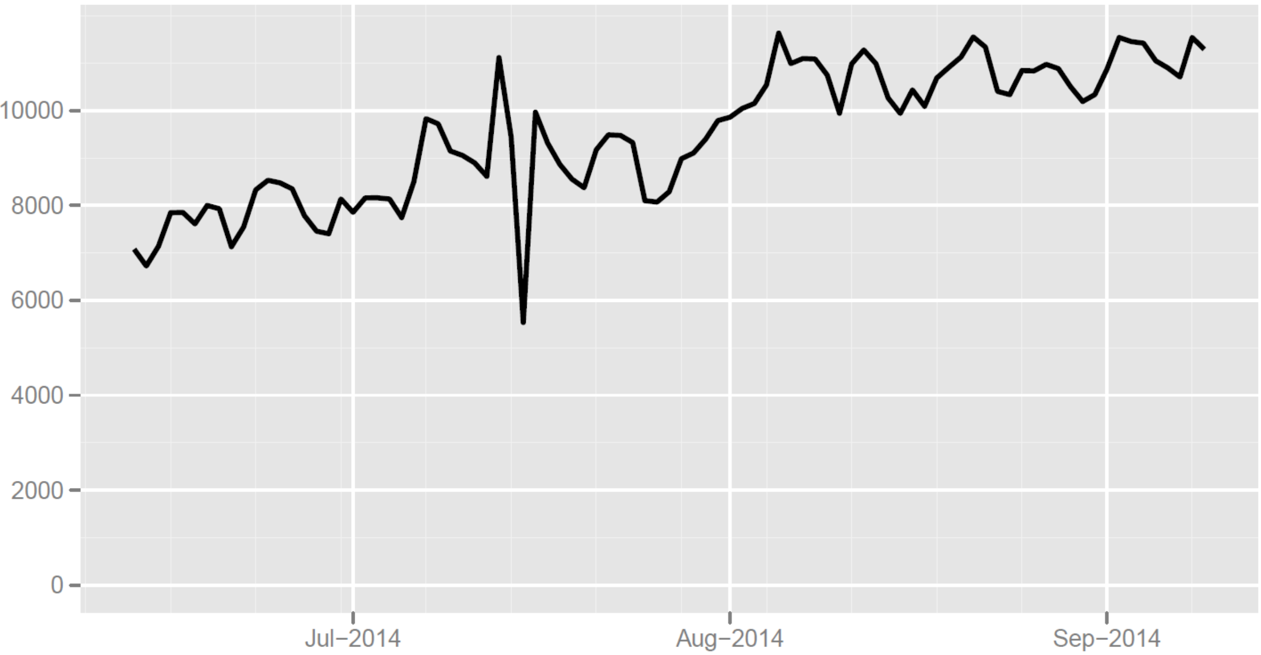

Directly connecting users

The Tor Project – https://metrics.torproject.org/

| Country | Mean daily users |
|---|---|
| United States | 333514 (14.28 %) |
| Germany | 206040 (8.82 %) |
| Russia | 161186 (6.90 %) |
| France | 140358 (6.01 %) |
| Brazil | 125856 (5.39 %) |
| Spain | 93393 (4.00 %) |
| United Kingdom | 89926 (3.85 %) |
| Italy | 85879 (3.68 %) |
| Poland | 65222 (2.79 %) |
| Argentina | 55872 (2.39 %) |

# Tor – Number of Bridge Users

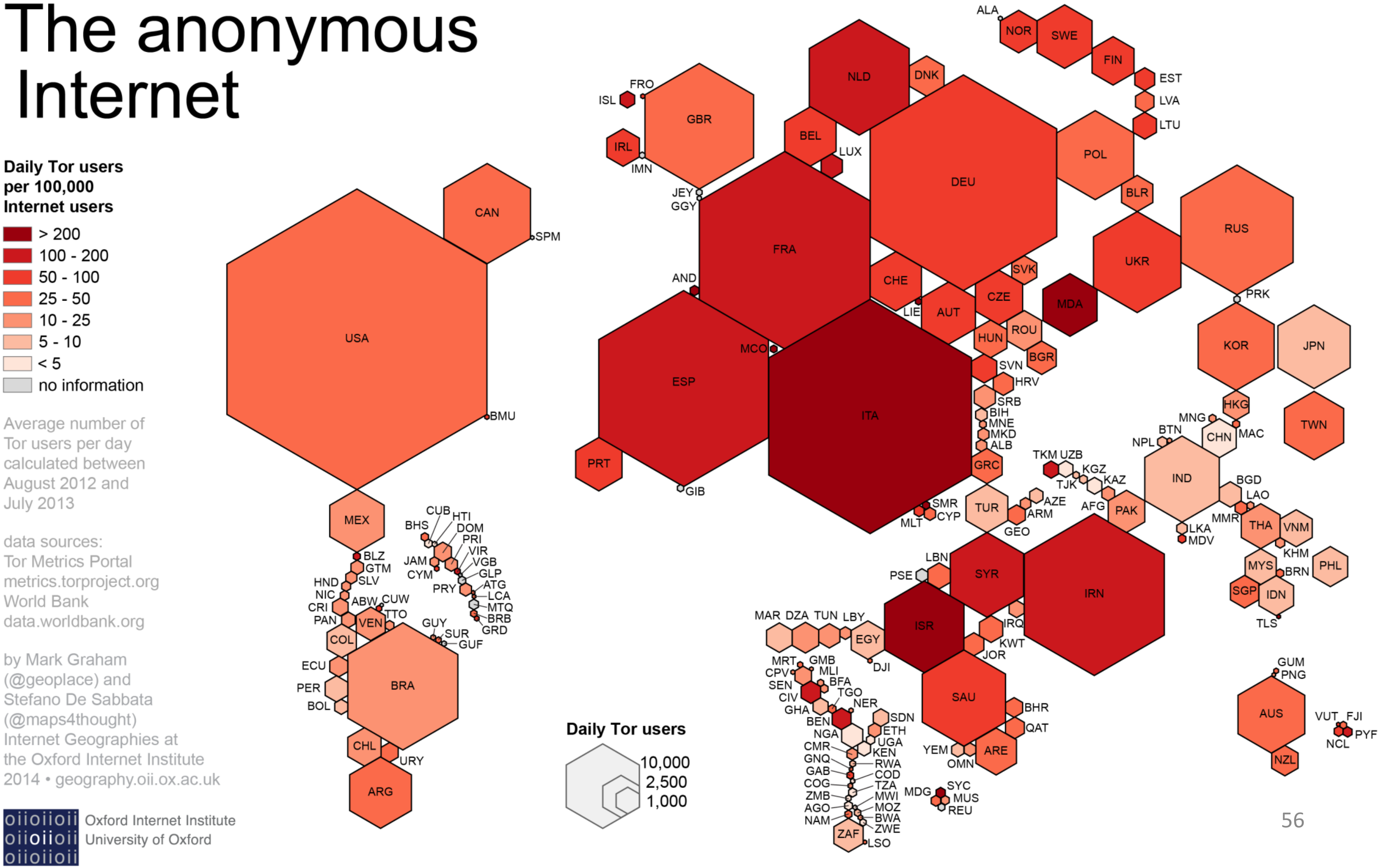
Bridge users



The Tor Project − https://metrics.torproject.org/

| Country | Mean daily users |
|---|---|
| Iran | 2086 (21.76 %) |
| United States | 945 (9.86 %) |
| Russia | 512 (5.34 %) |
| China | 353 (3.68 %) |
| United Kingdom | 349 (3.64 %) |
| India | 250 (2.61 %) |
| Iraq | 244 (2.55 %) |
| Saudi Arabia | 230 (2.40 %) |
| Germany | 219 (2.29 %) |
| Italy | 175 (1.83 %) |

# Tor - Globally



## The anonymous Internet

**Daily Tor users per 100,000 Internet users**

- > 200
- 100 - 200
- 50 - 100
- 25 - 50
- 10 - 25
- 5 - 10
- < 5
- no information

Average number of Tor users per day calculated between August 2012 and July 2013

data sources:
Tor Metrics Portal
metrics.torproject.org
World Bank
data.worldbank.org

by Mark Graham (@geoplace) and Stefano De Sabbata (@maps4thought)
Internet Geographies at the Oxford Internet Institute
2014 • geography.oii.ox.ac.uk

**Daily Tor users**

10,000
2,500
1,000

Oxford Internet Institute
University of Oxford

# Tor - Weaknesses

- Does not provide protection against end-to-end timing attacks
  - If the attacker can watch the traffic coming out of your computer, and also the traffic arriving at your chosen destination, he can use statistical analysis to discover that they are part of the same circuit
- Does not encrypt the traffic between an exit node and the target server
  - Any exit node is in a position to capture any traffic
  - Need to use SSL
- Incentive to participate as a bridge and keep a bridge alive/reliable
- Cat and mouse game between the censorship and researchers

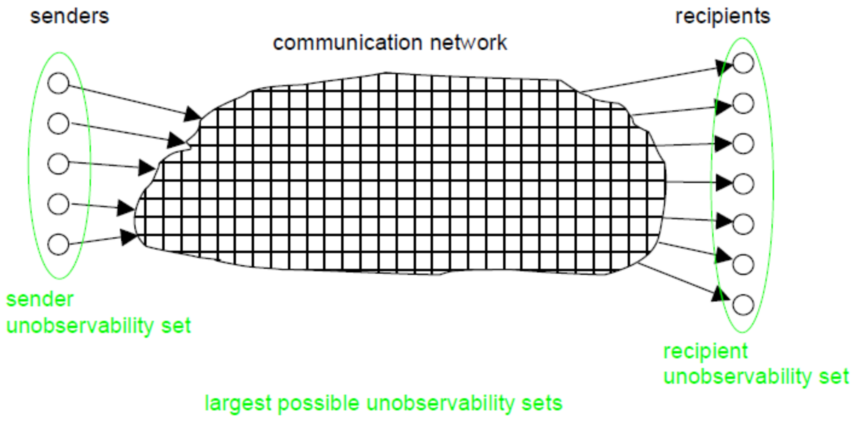
# Tor – Disclaimer (General to Anonymous Routing)

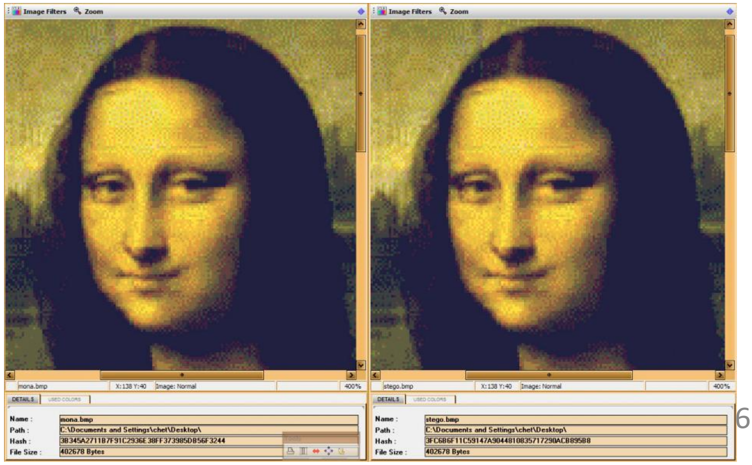
- *Criminals could in theory use Tor*
- Some responses:
  - – Criminals can already do bad things. Since they're willing to break laws, they already have lots of options available that provide *better* privacy than Tor provides
  - – Tor aims to provide protection for ordinary people who want to follow the law
  - – Some advocates of anonymity explain that it's just a tradeoff — accepting the bad uses for the good ones

# Systems for Unobservability

# Reminder - Unobservability

- Unobservability is the state of items of interest (IOIs) being indistinguishable from any IOI (of the same type)
- Sender unobservability means it is not noticeable whether any sender within the unobservability set sends
- Recipient unobservability means it is not noticeable whether any recipient within the unobservability set receives
- Desirable property of steganographic systems
- Related to Oblivious RAM and Private Information Retrieval (PIR)


senders
communication network
recipients
sender unobservability set
recipient unobservability set
largest possible unobservability sets



mona.bmp X:138 Y:40 Image: Normal 400%

Name : mona.bmp
Path : C:\Documents and Settings\chet\Desktop\
Hash : 3B345A2711B7F91C2936E38FF373985DB56F3244
File Size : 402678 Bytes

stego.bmp X:138 Y:40 Image: Normal 400%

Name : stego.bmp
Path : C:\Documents and Settings\chet\Desktop\
Hash : 3FC6B6F11C59147A9044810835717290ACB89588
File Size : 402678 Bytes

# Reminder
# Anonymity vs. Unobservability

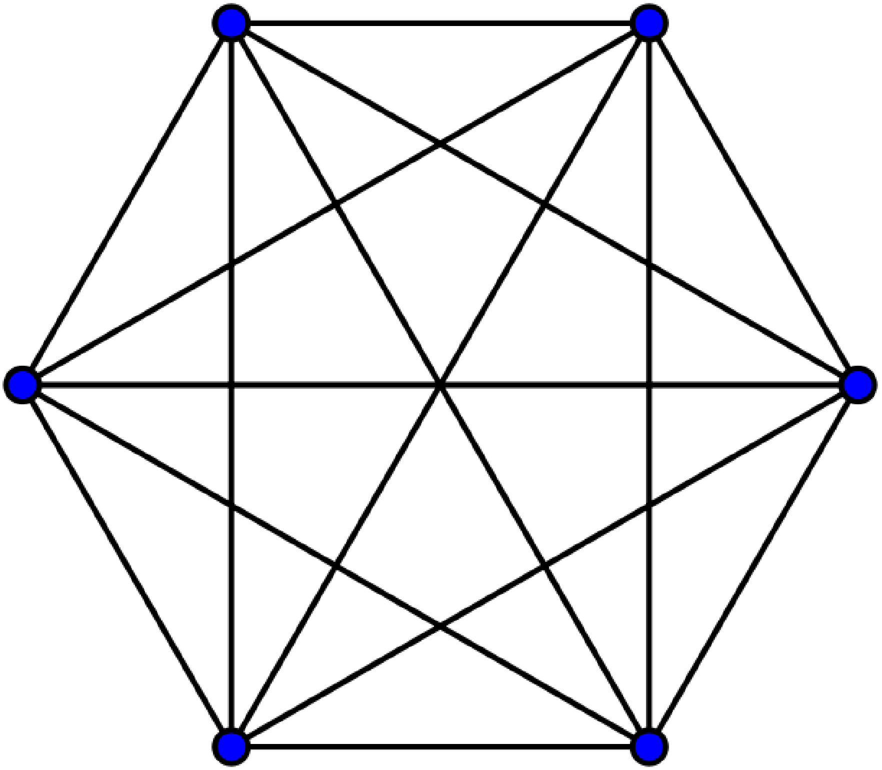
- Unobservability implies anonymity
- Anonymity does not imply unobservability
  - Anonymity only hides the identity of the sender/receiver, it does not guarantee unobservability

# Systems for Unobservability

- Anonymity systems:
  - Prevent an attacker from determining who is communicating with whom
  - Adversary can still identify which senders or recipients were active during a period of observation
- Anonymity systems that provide *unobservability:*
  - An adversary monitoring the users is unable to distinguish messages carrying actual content from random noise
  - Hide which users sent or received a message during a period of observation
  - Adds latency and overhead

# Systems for Unobservability – Naïve Solution

- All *n* users in the system simultaneously broadcast a fixed-length message of *l* bytes
  - Everybody send to everybody at the same time
  - One or more can be real communication
  - Inefficient in terms of communication overhead



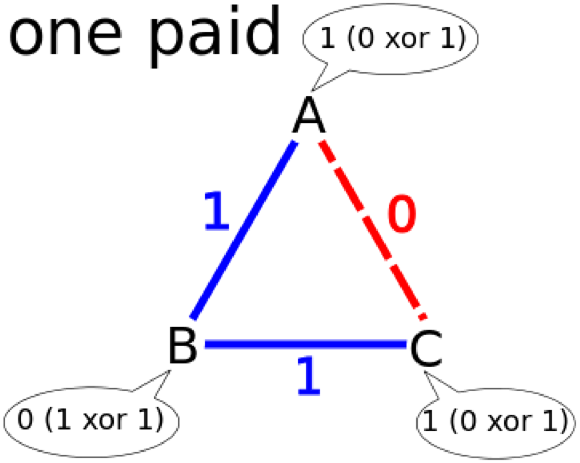
# Dining Cryptographers Problem

- Introduced by David Chaum, 1988
- Problem statement:
  - Group of cryptographers seated around a table having dinner
  - They are informed by the waiter that their bill has already been paid for anonymously
  - They speculate that their bill was either paid by one of them, or by the NSA
  - They want to know if NSA in fact paid the bill
  - If one of the cryptographers did, in fact, pay the bill, can he let the others know while remaining anonymous?
- Solution:
  - Each cryptographer secretly flips a coin and records a 0 for tails and 1 for heads
  - Cryptographers compute $z = x \oplus y$
    - $x$ is the result of his or her own flip
    - $y$ is the value learned from the cryptographer to his right
  - He announces the result to the group
  - If one of the cryptographers paid for the meal, he instead announces $z \oplus 1$
  - They can all jointly compute $Z = z_1 \oplus z_2 \oplus \cdots \oplus z_n$
  - If $Z = 1$, then one of the cryptographers at the table paid the bill

# Dining Cryptographers Problem

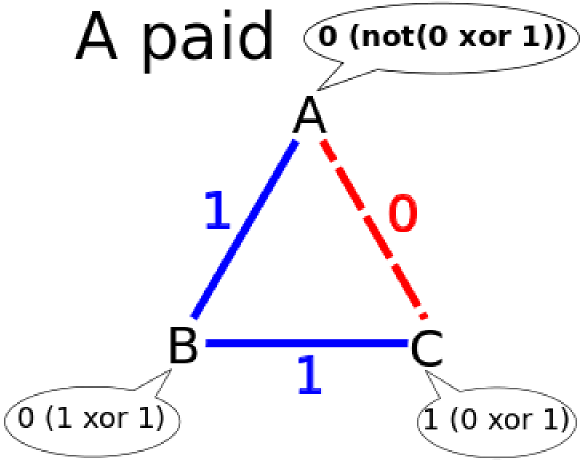
- Let $x_{i,j}$ be the bit shared between neighboring cryptographers $i$ and $j$
- Let $s_i$ be cryptographer $i$'s secret bit
- Each cryptographer $i$ announces the result of $z_i = x_{(i-1),i} \oplus x_{i,(i+1)} \oplus s_i$
- $Z = z_1 \oplus z_2 \oplus \cdots \oplus z_n = s_1 \oplus s_2 \oplus \cdots \oplus s_n$

None paid



1 xor 1 xor 0 = 0

A paid



1 xor 0 xor 0 = 1

# DC Networks

- Goal: How can  a member of the network transmit a bit without any other communicant knowing who transmitted it?
- Offers information-theoretically secure anonymity
- Bandwidth overhead: all users send the result of their local computation to every other user
  - Total of $n(n - 1)$ bits are transmitted in order to anonymously send a single message bit
- Message collisions: two senders trying to transmit messages at the same time
- Security: two nodes can collude to reveal the message sent by a node between them

# Dissent [1]

- **Group communication** model
  - Builds on dining cryptographers and verifiable shuffle algorithms to offer provable anonymity guarantees
  - Wish to hide *which member* sent a message, but make it clear that *some member* sent it

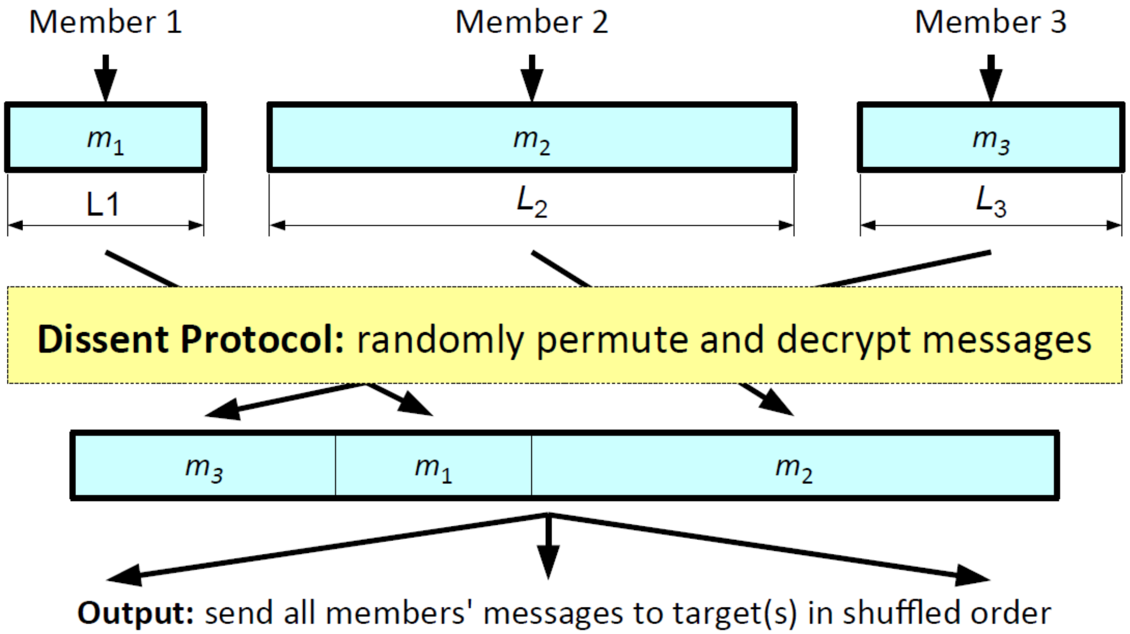

Member 1
Member 2
Member 3
$m_1$
$m_2$
$m_3$
L1
$L_2$
$L_3$
**Dissent Protocol:** randomly permute and decrypt messages
$m_3$
$m_1$
$m_2$
**Output:** send all members' messages to target(s) in shuffled order

Figure: Bryan Ford

[1] Dissent: Accountable Group Anonymity, Henry Corrigan-Gibbs and Bryan Ford. CCS 2010

# Herbivore

- Hierarchical DC-net
- When a new user joins the network, it is assigned to one of many smaller groups of users called *cliques*
- Every clique has between *k* and *3k* users, where *k* is a parameter of the system
- Anonymous slot-reservation protocol for collision avoidance in a DC-net
    - Nodes reserve bandwidth on the channel and then transmit fixed blocks of data

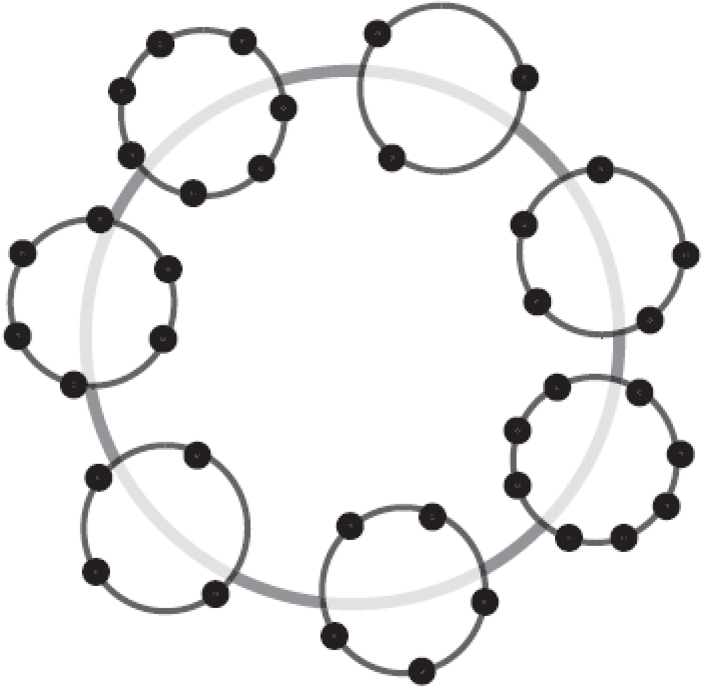


Figure: Goel et al. Cornell Univ. CIS Tech. Rep., 2003

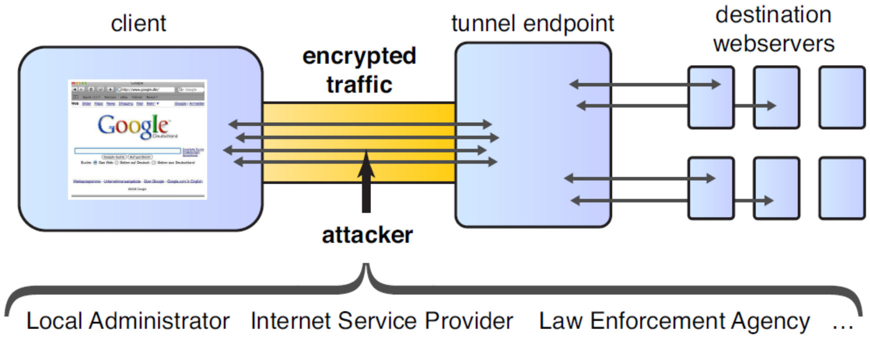
GOEL, S., ROBSON, M., POLTE, M., AND SIRER, E. G. 2003. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Tech. rep. 2003-1890. Cornell University, Ithaca, NY. February.

# Other Deployed Systems

- $\mathcal{P}^5$
  - A user first encrypts the message with the intended recipient's public key and then broadcasts the ciphertext to one of the broadcast groups the sender has joined
  - Users also generate noise traffic at a constant rate
    - Noise packets are indistinguishable from an authentic message packet
- Nonesuch
  - Users use steganography to hide a message in an image file, and then post that image file to a Usenet newsgroup
  - Other images in the newsgroup serve as cover traffic
  - Nonesuch nodes periodically retrieve all news images that were posted to the group and try to extract a stegotext from each
  - Hidden messages are encrypted

# Website Fingerprinting

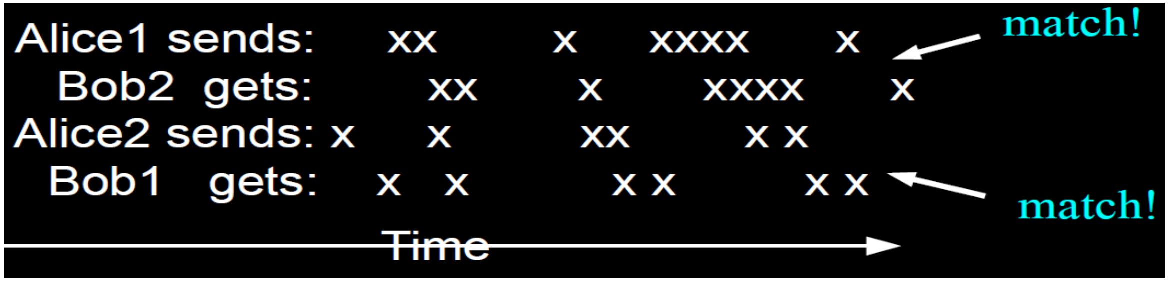
- Inferring the Website that a target user is browsing
- Adversary observes the quantity and length of data packets received when browsing on various websites
- Adversary builds a fingerprint of what the website's response traffic looks like when fetched via an encrypted connection
- Adversary matches a user's traffic to these fingerprints
- Mitigation:
  - Split messages into fixed length cells
  - Ensure all transmitted packets are of the same length



**Example:** Encrypted link (tunnel) to a trusted server located on the Internet. *Tunnel endpoint* relays the HTTP requests of one or multiple clients to the various destination web servers

# Timing Attacks

- A passive global adversary who is able to observe connections entering and exiting the anonymity network may link inputs and outputs based on their patterns of packet inter-arrival times



- Mitigation:
  - *defensive dropping*: nodes introduce random packets called *dummy traffic* at the start of the circuit
  - dummy traffic is probabilistically dropped as it passes through the network
- An active attacker can induce timing delays into the client’s traffic that allows him to easily correlate input and output flows in an anonymity network

# Predecessor Attacks

- Identify the initiator of a particular persistent connection
- If an adversary can observe the first and last hop in a client’s circuit, he can perform a timing analysis attack to try to link the two - Syverson et al. (2000)
  - Circuit’s initiator is the immediate predecessor of the adversary observing the client’s entry node
  - *(C/N)*$^2$ bound for the adversary to succeed - remember Tor entry guards
- Expected number of rounds (path reformations) it takes an adversary to be able to identify the initiator of a connection with high confidence - Wright et al. (2004)
  - Crowds, onion routing, mix networks, DC-nets
  - DC-net offers the best resilience against the predecessor attack

# Disclosure/Intersection Attacks

- Alice repeatedly sends messages to one of $m$ different communication partners in each mix round
- A passive adversary observes the messages entering and exiting the mix and wants to identify with whom Alice is corresponding
- Adversary first waits until he has observed $m$ mutually disjoint set of recipients $\{R_1, \ldots, R_m\}$
  - Each of these sets contains precisely one of Alice's $m$ communication partners
- Adversary refines these sets by waiting to observe a new recipient set $R$ that intersects with *only one* previously observed $R_i$
  - Assuming during rounds $R$ and $R_i$ Alice communicated with the same partner
- $R_i$ is then reduced to $R_i \cap R$
- Adversary can continue this attack until he has reduce all *Ri* to only a single element
- No efficient method for absolutely preventing intersection attacks

# Conclusion

- Lots of research over the last 3 decades on anonymous routing in the Internet
- High-latency systems (now abandoned)
- Low-latency systems
  - End-to-end circuit with onion routing
  - Prominent example: Tor
  - Still vulnerable to some adversaries
- Unobservability
  - Dining cryptographers as starting point

# References

- M. Edman and B. Yener. On anonymity in an electronic society: A survey of anonymous communication systems. ACM Computing Surveys, 42(1), 2009
- P. Golle and A. Dining cryptographers revisited. In Proceedings of Eurocrypt 2004.
- R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation Onion router. In Proceedings of 13th USENIX Security Symposium, 2004.
- https://www.torproject.org