A SIMPLE INTRODUCTION TO TOR

The Onion Router

Fabrizio d'Amore

Privacy on Public Networks

- Internet is designed as a public network
  - Wi-Fi access points, network routers see all traffic that passes through them
- Routing information is public
  - IP packet headers identify source and destination
  - Even a passive observer can easily figure out who is talking to whom
- Encryption does not hide identities
  - Encryption hides payload, but not routing information
  - Even IP-level encryption (tunnel-mode IPsec/ESP) reveals IP addresses of IPsec gateways

Anonymity

- Anonymity = the person is not identifiable within a set of subjects
  - You cannot be anonymous by yourself!
    - Big difference between anonymity and confidentiality
  - Hide your activities among others' similar activities
- Unlinkability of action and identity
  - For example, sender and his email are no more related after adversary's observations than they were before
- Unobservability (hard to achieve)
  - Adversary can't even tell whether someone is using a particular system and/or protocol

Attacks on Anonymity

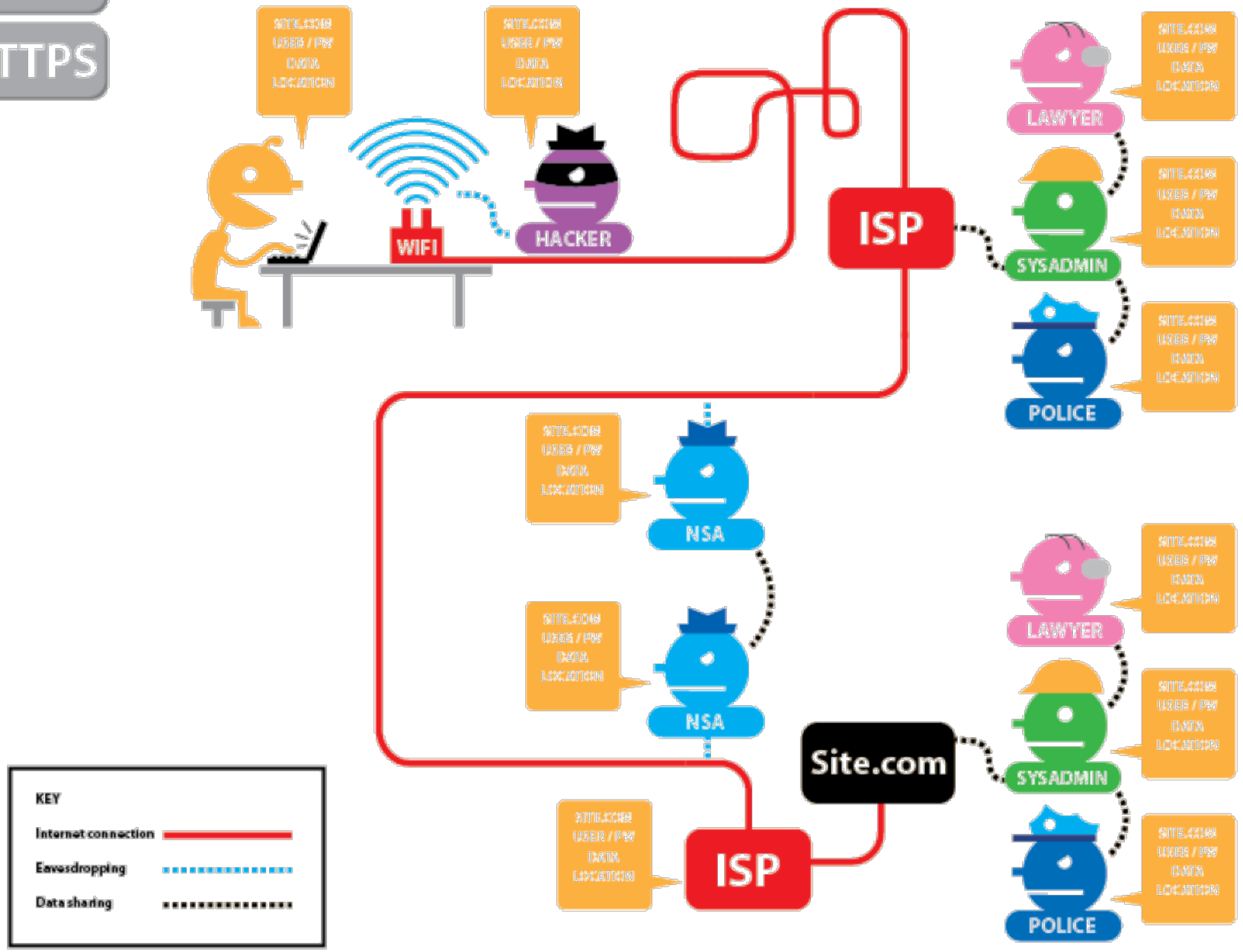
- Passive traffic analysis
  - Infer from network traffic who is talking to whom
- Active traffic analysis
  - Inject packets or put a timing signature on packet flow
- Compromise of network nodes
  - Attacker may compromise some routers
  - It is not obvious which nodes have been compromised
    - Attacker may be passively logging traffic
  - Better not to trust any individual router
    - Can assume that some fraction of routers is good, but don’t know which

Tor

- Deployed onion routing network
  - http://torproject.org
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
  - Thousands of relay nodes, 100K-500K? of users
- Easy-to-use client proxy, integrated Web browser
- Tor is a distributed anonymous communication service using an overlay network that allows people and groups to improve their privacy and security on the Internet.
- Individuals use Tor to keep websites from tracking them, or to connect to those internet services blocked by their local Internet providers.
- Tor's hidden services let users publish web sites and other services without needing to reveal the location of the site.

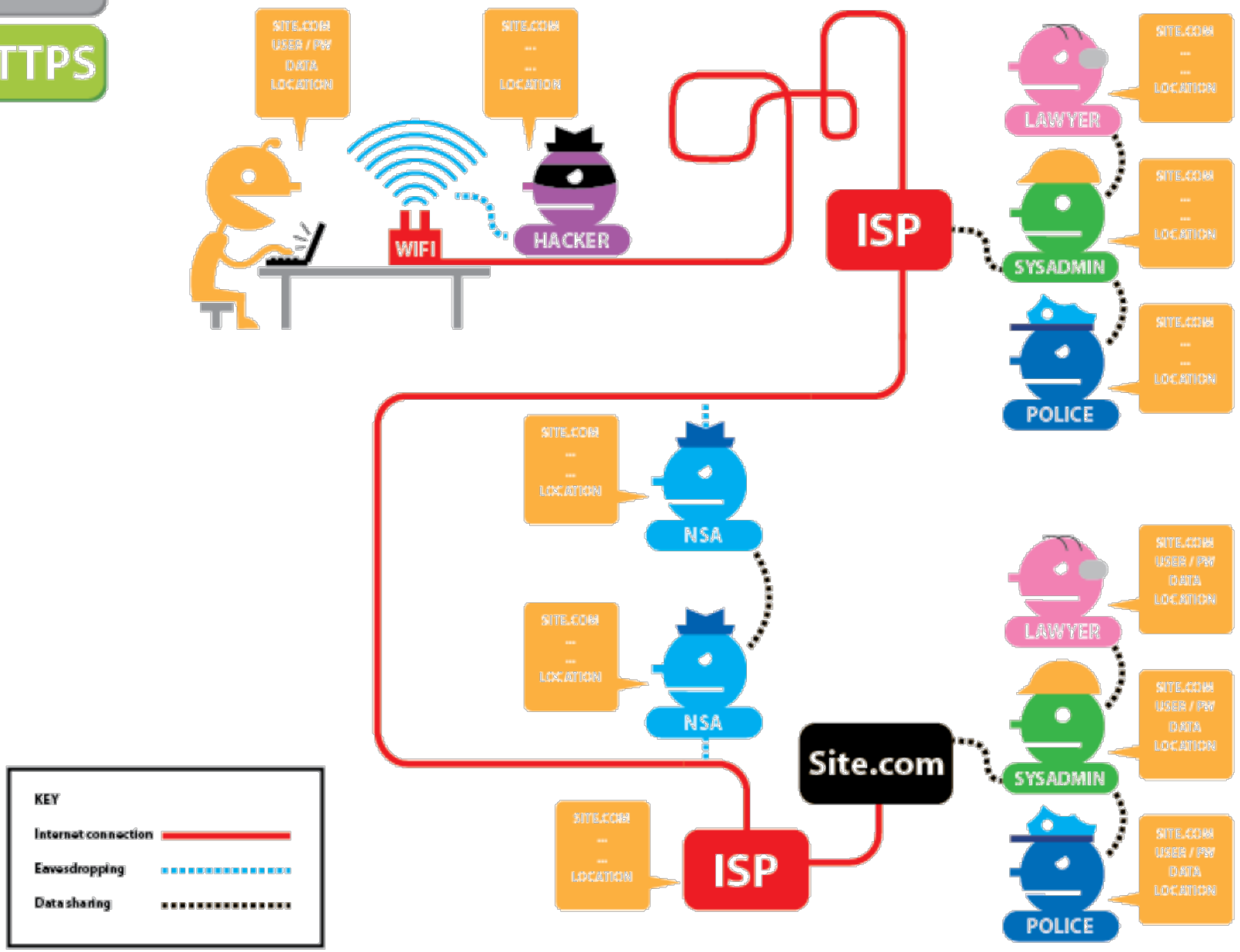
Tor

HTTPS


SITE.COM
USER / PW
DATA
LOCATION
WIFI
HACKER
ISP
LAWYER
SYSADMIN
POLICE
NSA
NSA
Site.com
ISP
KEY
Internet connection
Eavesdropping
Data sharing

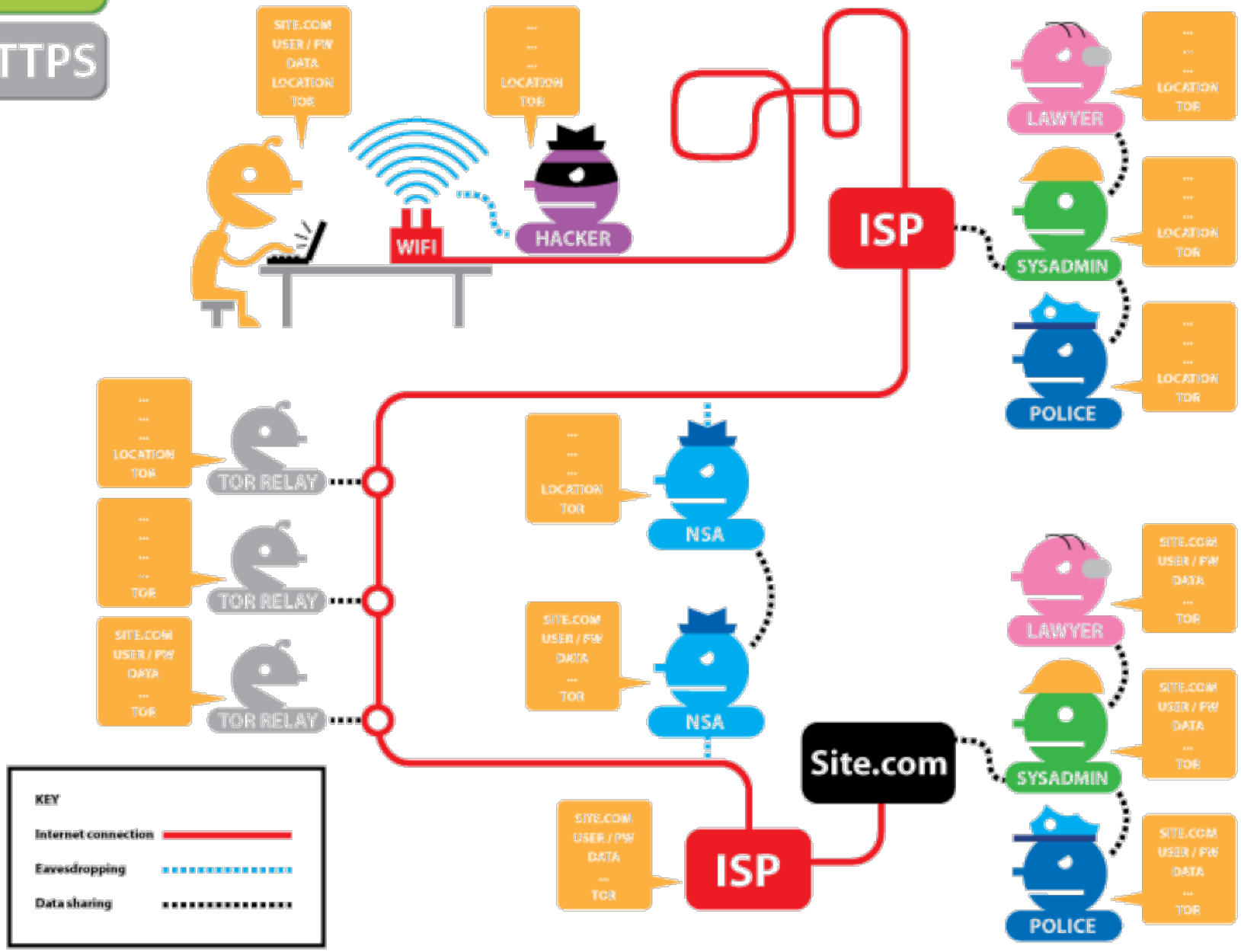
Tor

HTTPS


WIFI
HACKER
ISP
LAWYER
SYSADMIN
POLICE
NSA
NSA
Site.com
ISP
LAWYER
SYSADMIN
POLICE
KEY
Internet connection
Eavesdropping
Data sharing

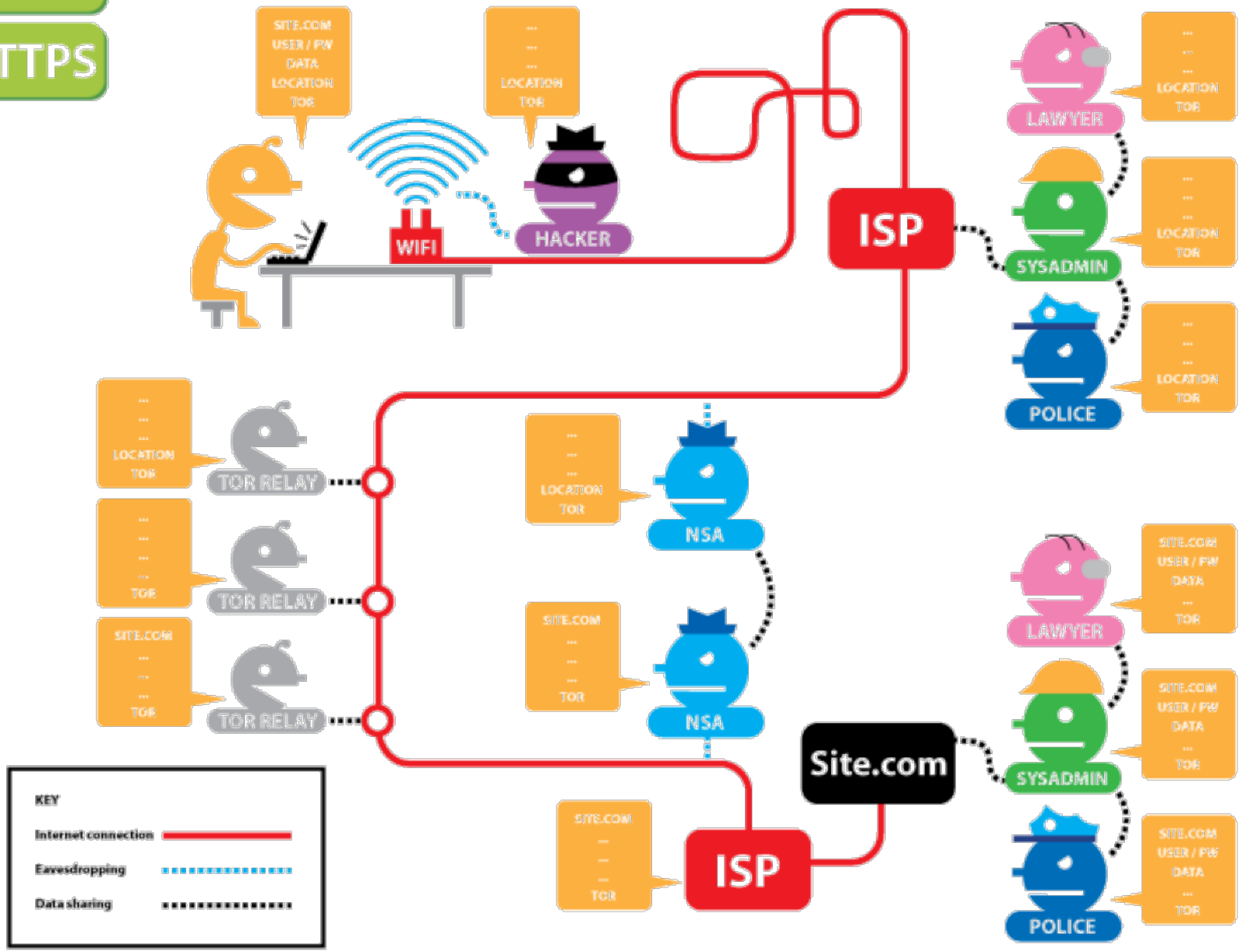
Tor

HTTPS

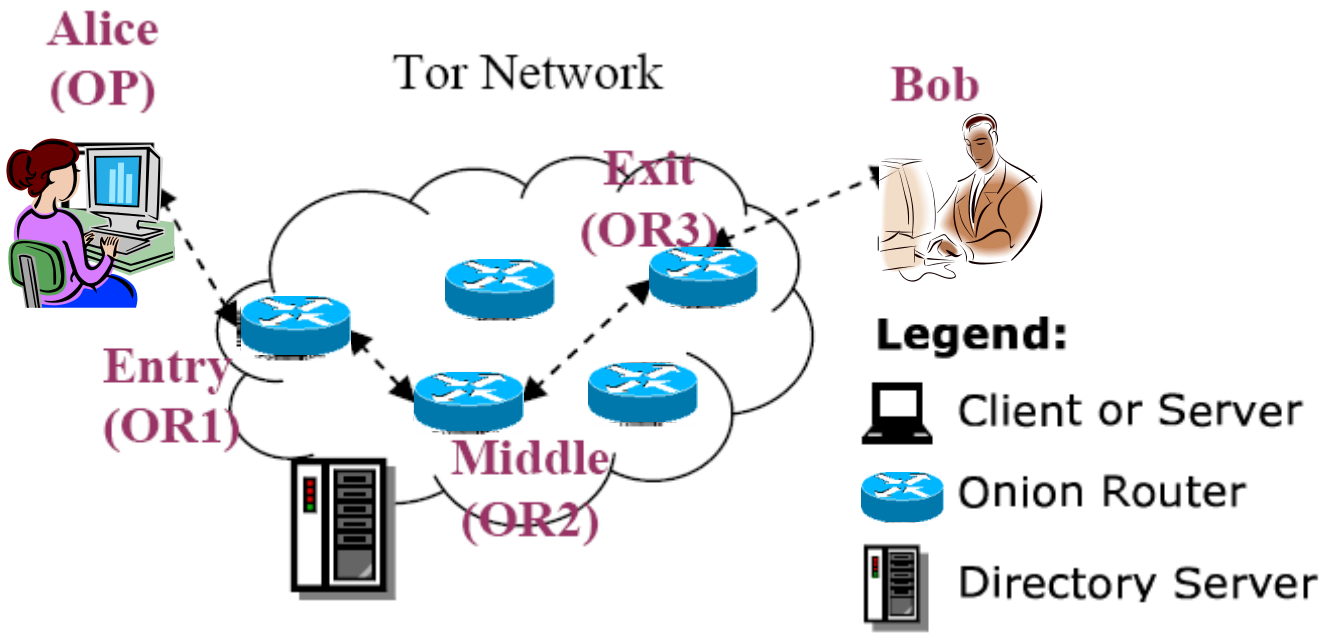

SITE.COM
USER / PW
DATA
LOCATION
TOR
...
LOCATION
TOR
WIFI
HACKER
ISP
LAWYER
SYSADMIN
POLICE
TOR RELAY
TOR RELAY
TOR RELAY
NSA
NSA
Site.com
ISP
KEY
Internet connection
Eavesdropping
Data sharing

Tor

HTTPS

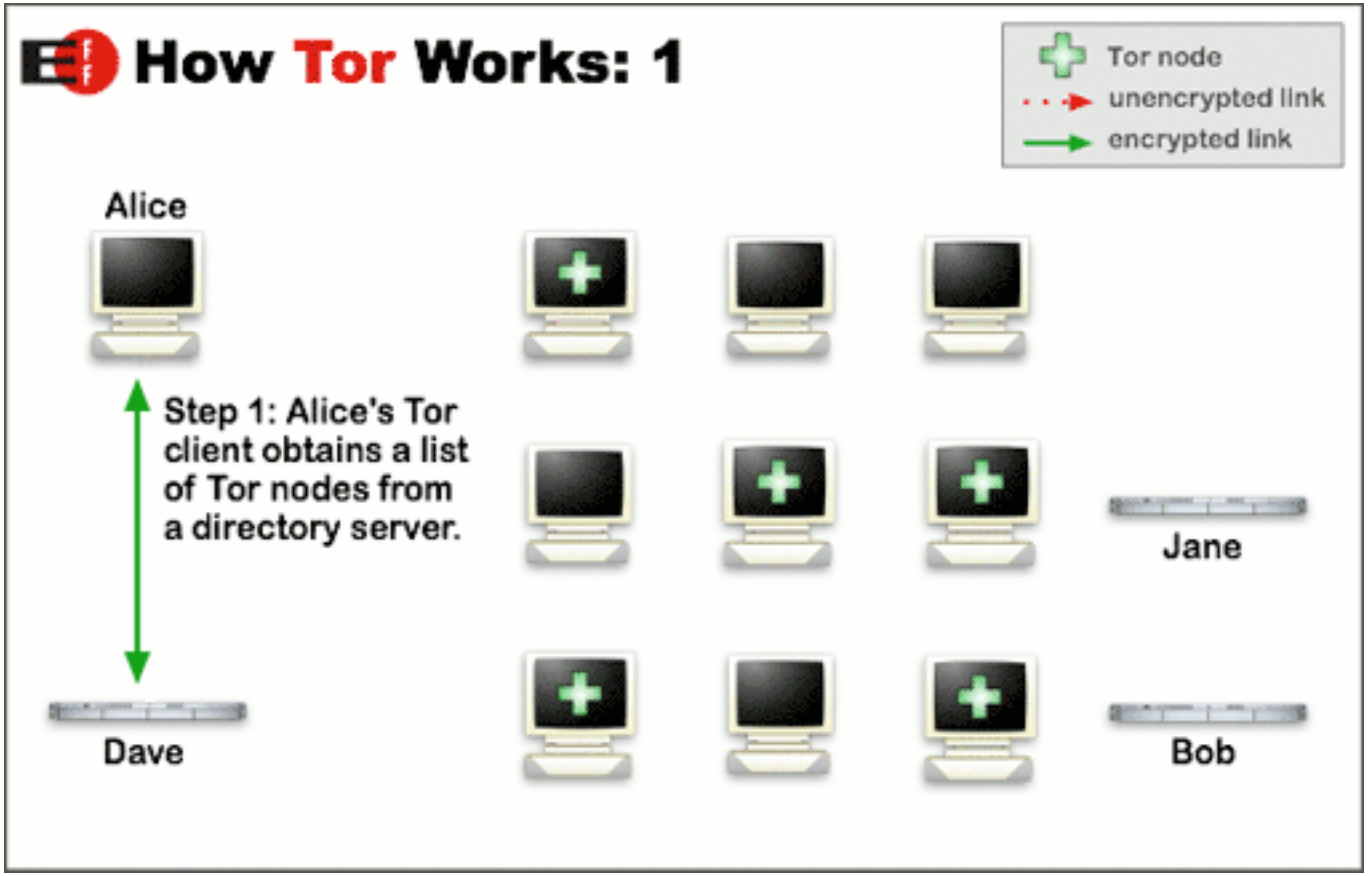

SITE.COM
USER / PW
DATA
LOCATION
TOR
LOCATION
TOR
WIFI
HACKER
ISP
LAWYER
SYSADMIN
POLICE
LOCATION
TOR
TOR RELAY
TOR RELAY
TOR RELAY
NSA
NSA
SITE.COM
USER / PW
DATA
TOR
Site.com
ISP
KEY
Internet connection
Eavesdropping
Data sharing

Components of Tor

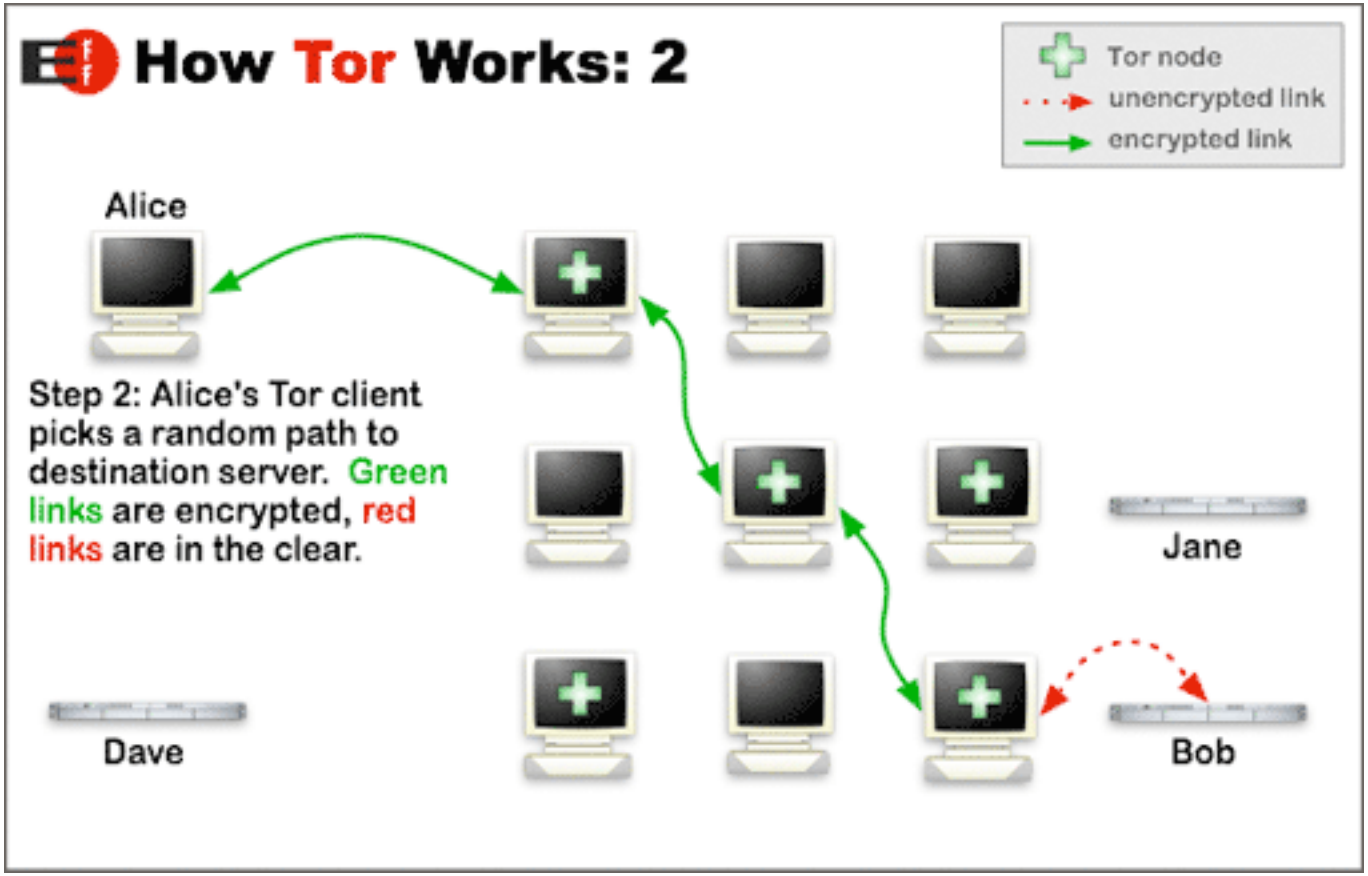


- **Client**: the user of the Tor network
- **Server**: the target TCP applications such as web servers
- **Tor (onion) router**: the special proxy relays the application data
- **Directory server**: servers holding Tor router information

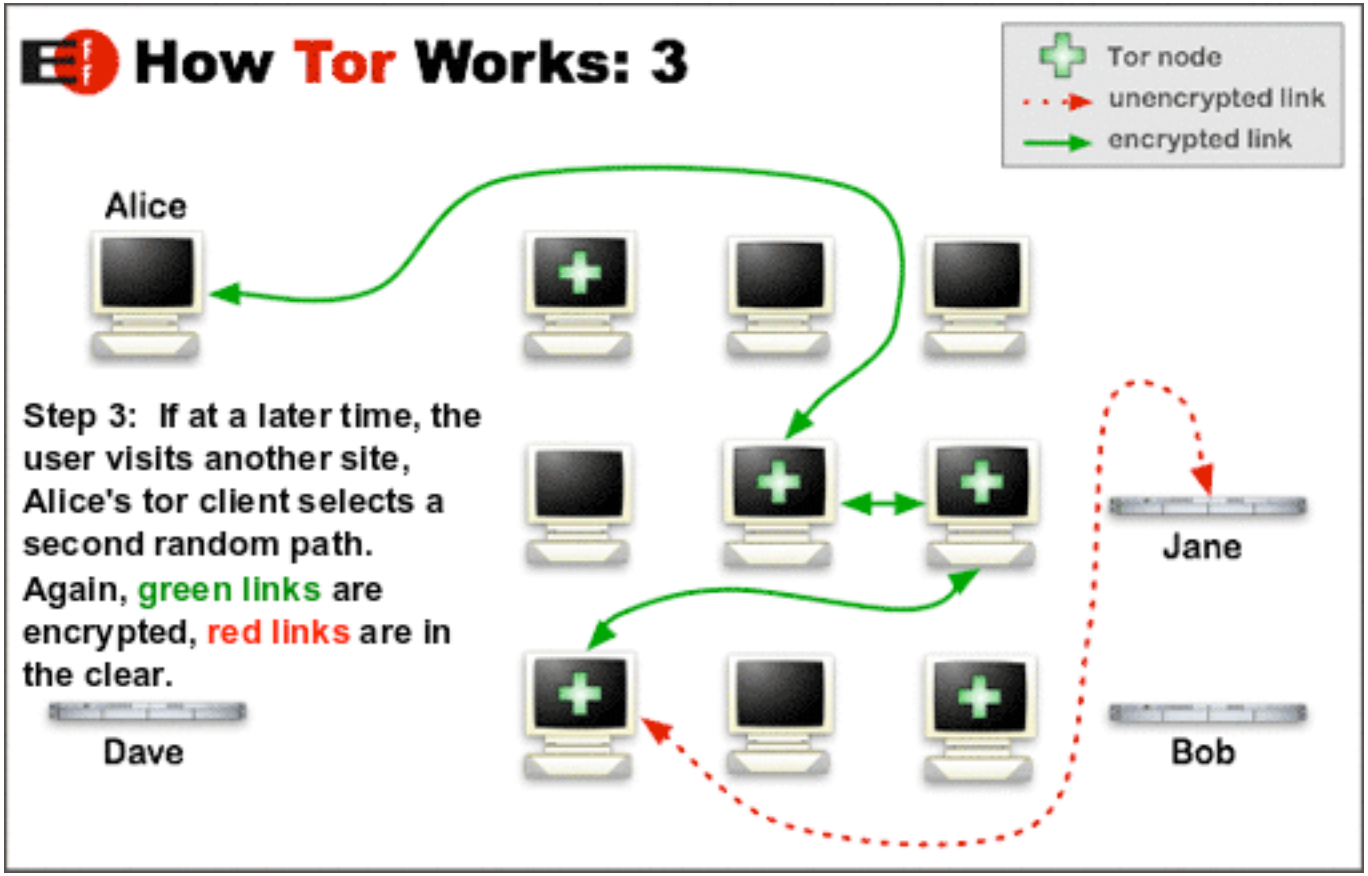
Tor operations 1



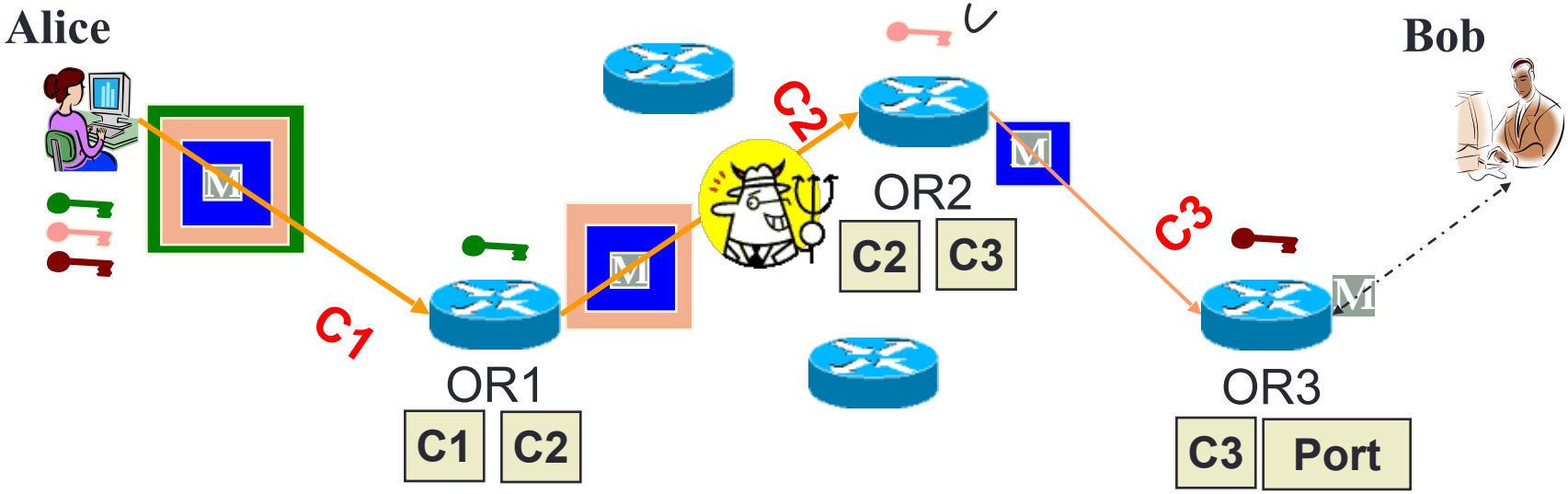
Tor operations 2



Tor operations 3

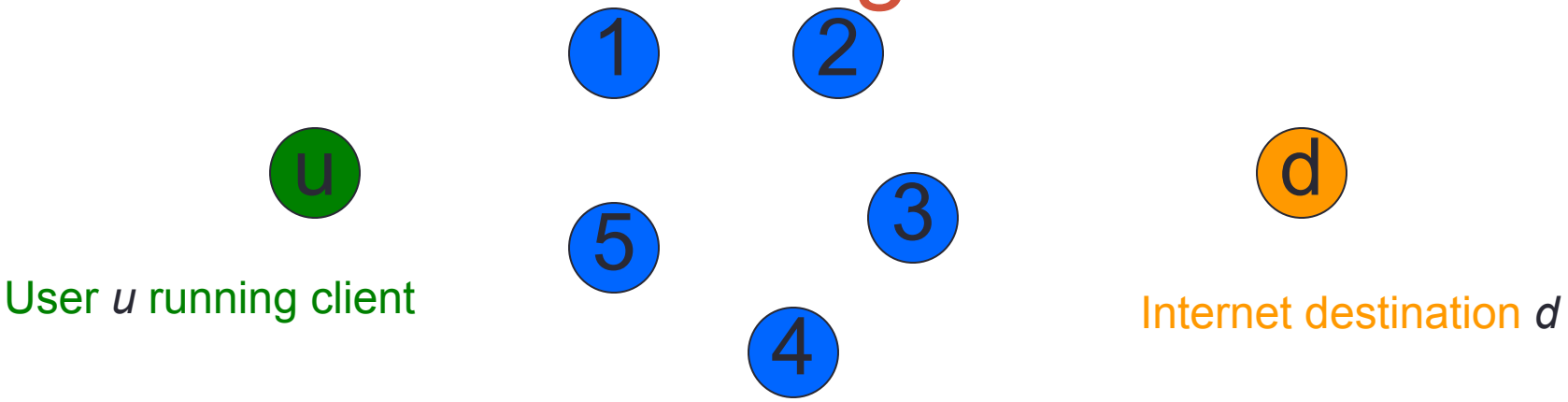


How Tor Works? --- Onion Routing

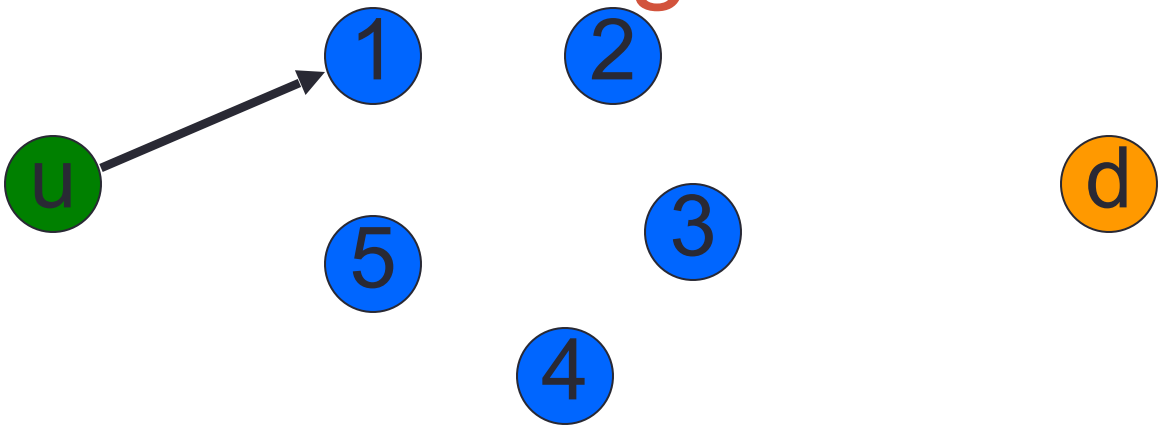


- A circuit is built incrementally one hop by one hop
- Onion-like encryption
  - Alice negotiates an AES key with each router
  - Messages are divided into equal sized cells
  - Each router knows only its predecessor and successor
  - Only the Exit router (OR3) can see the message, however it does not know where the message is from

How Onion Routing Works

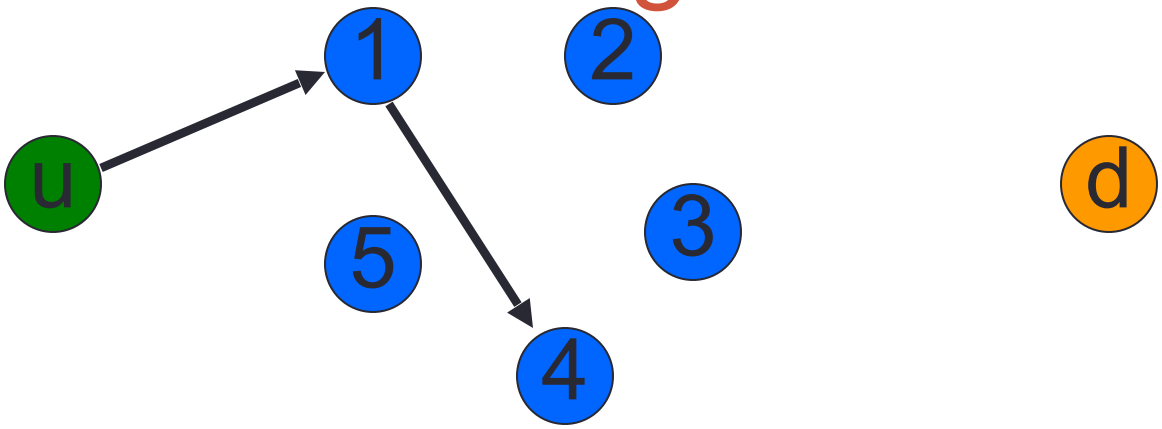


How Onion Routing Works



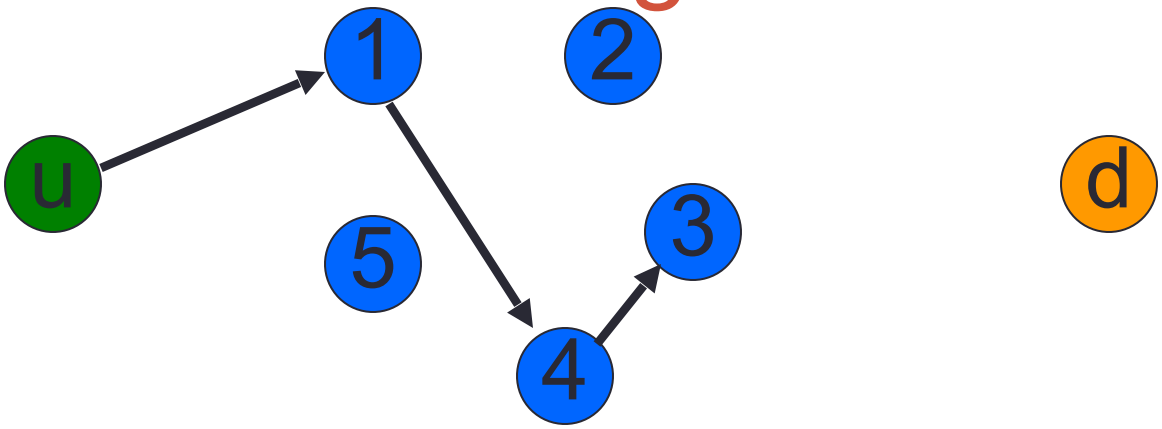
1. *u* creates *l*-hop circuit through routers

How Onion Routing Works



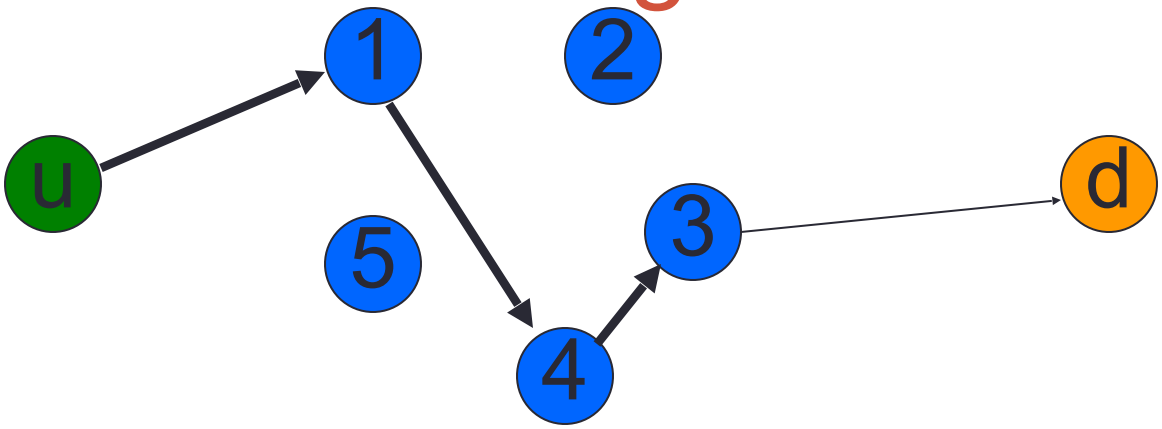
1. *u* creates *l*-hop circuit through routers

How Onion Routing Works



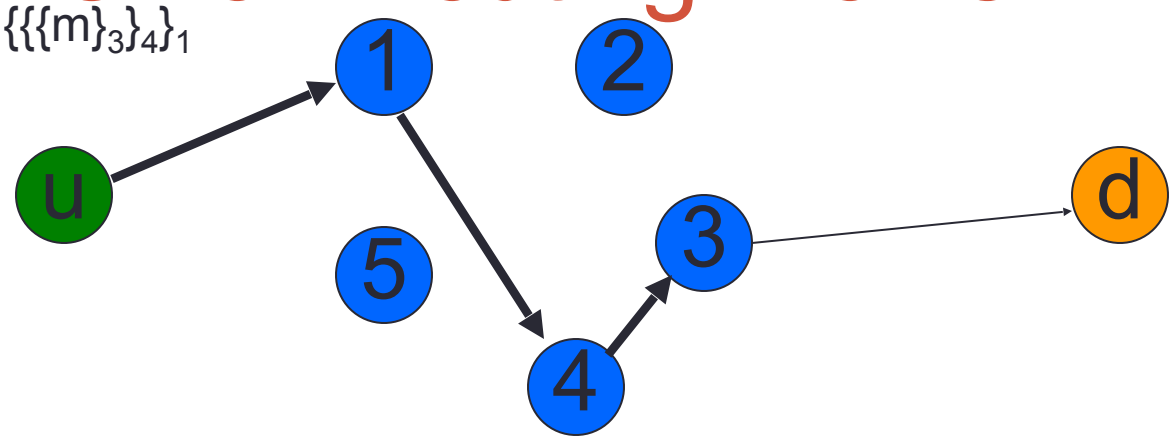
1. *u* creates *l*-hop circuit through routers

How Onion Routing Works



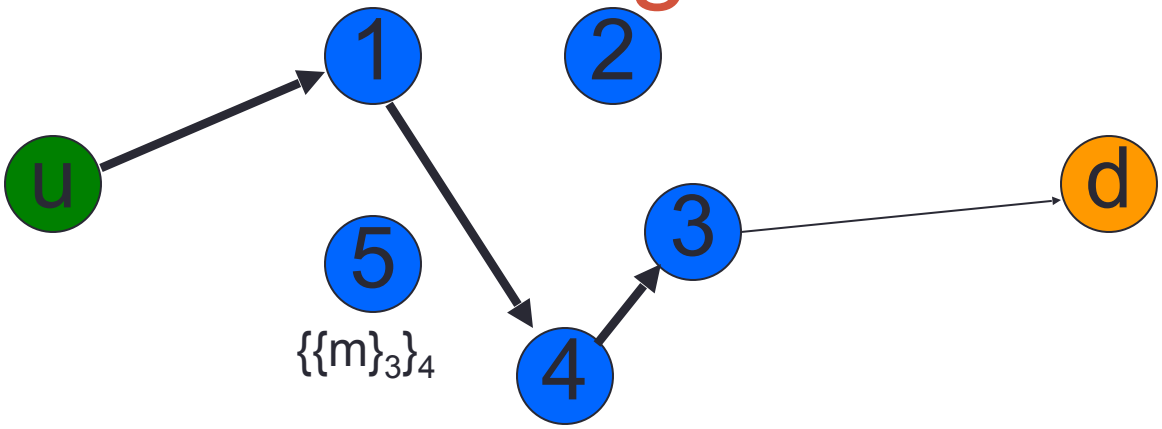
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*

How Onion Routing Works



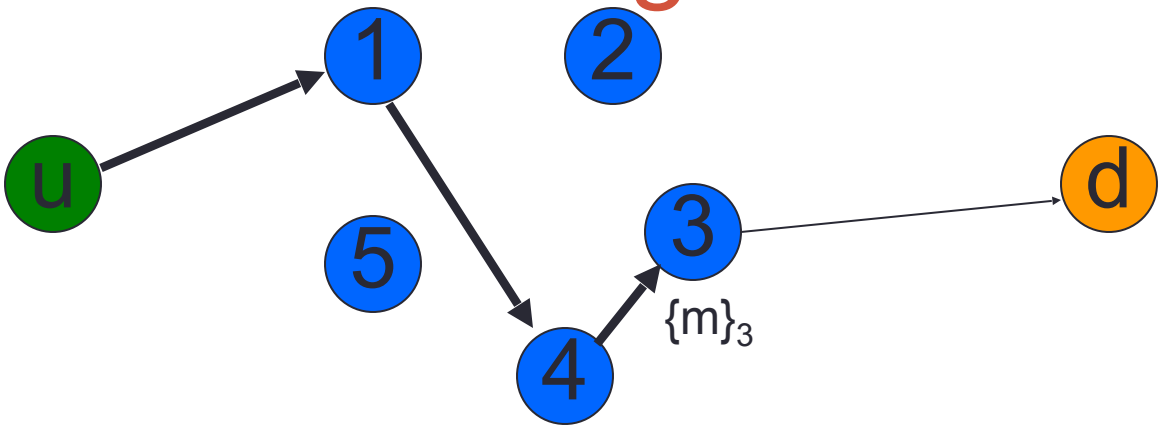
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



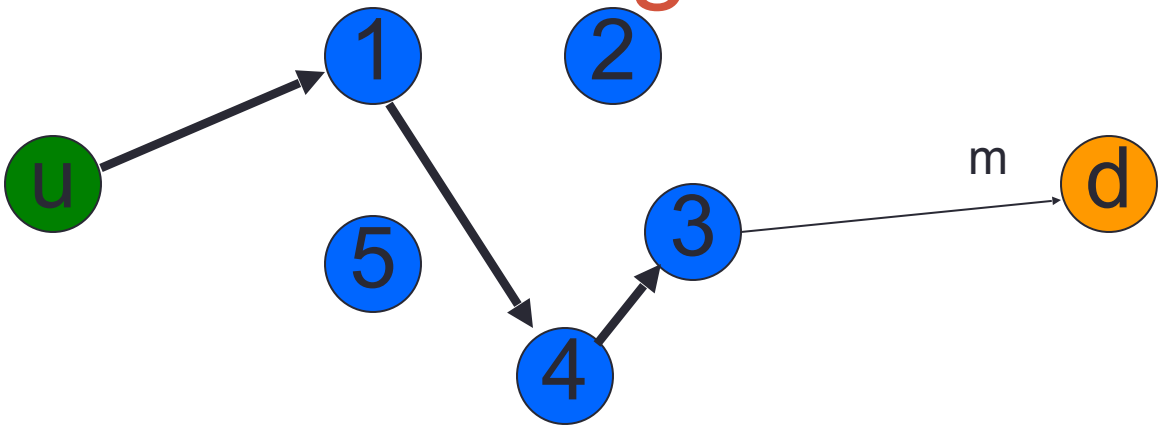
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



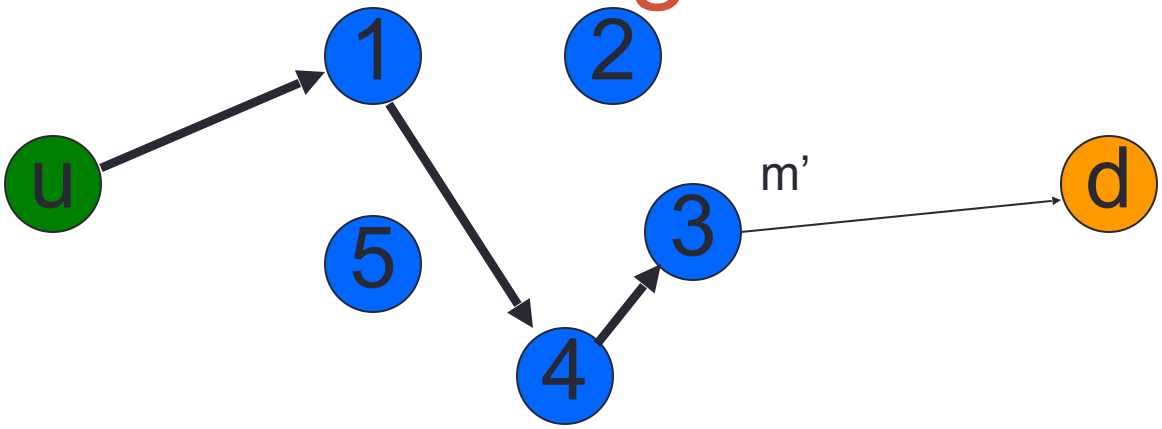
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



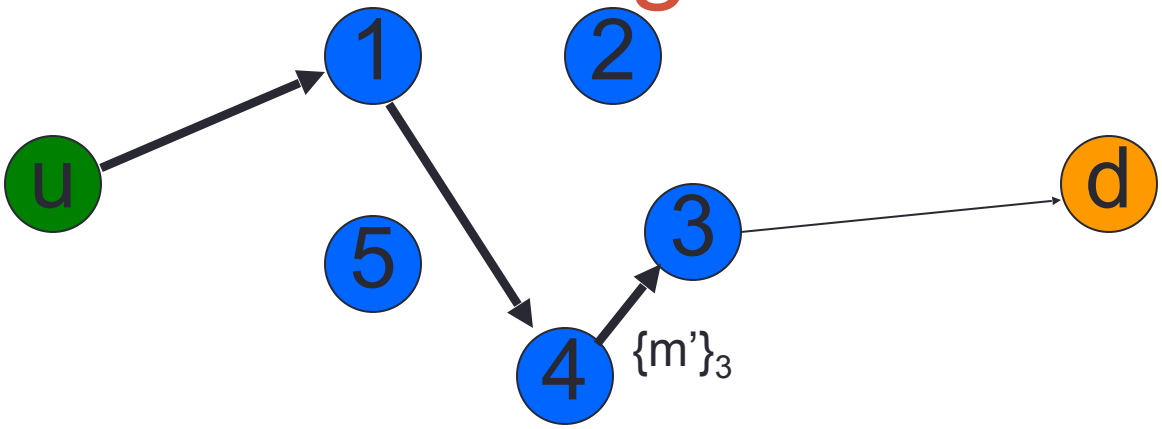
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



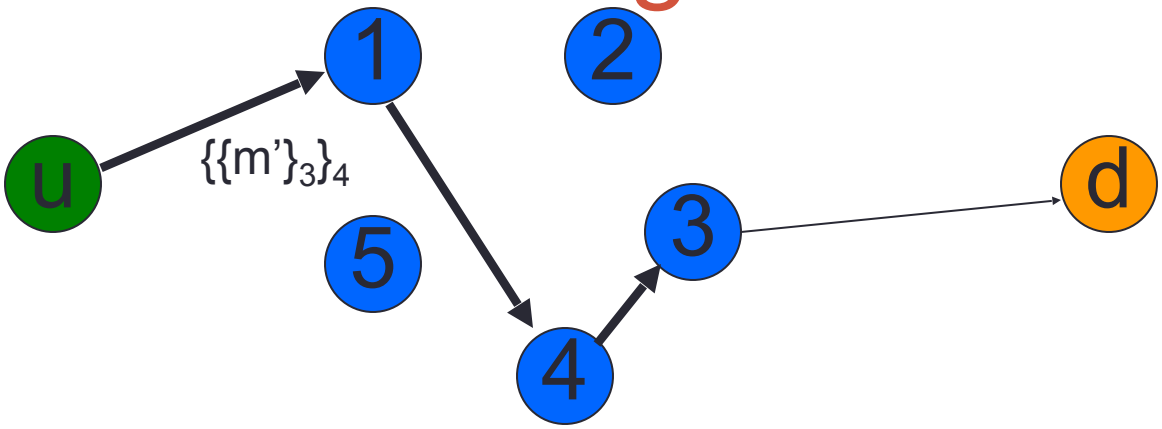
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



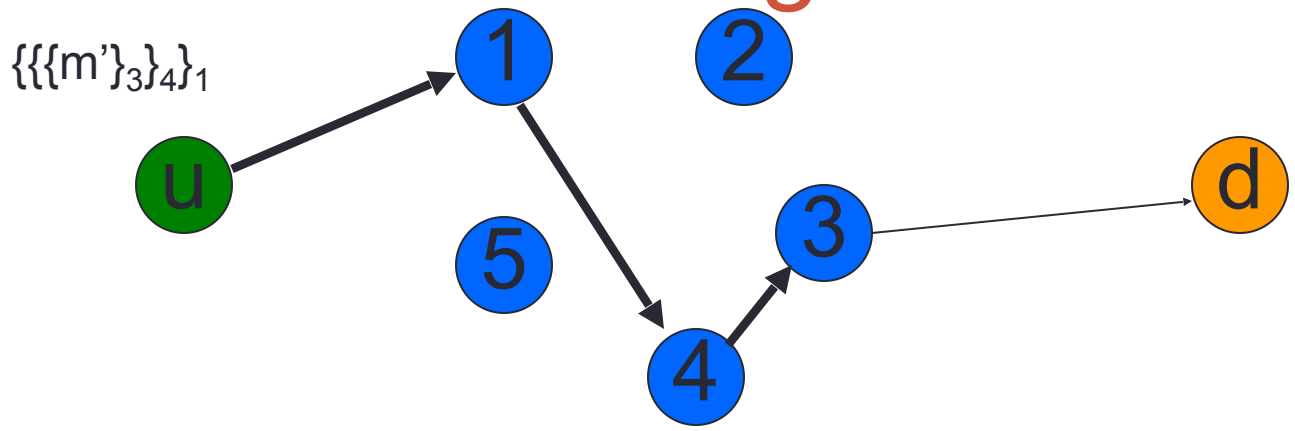
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



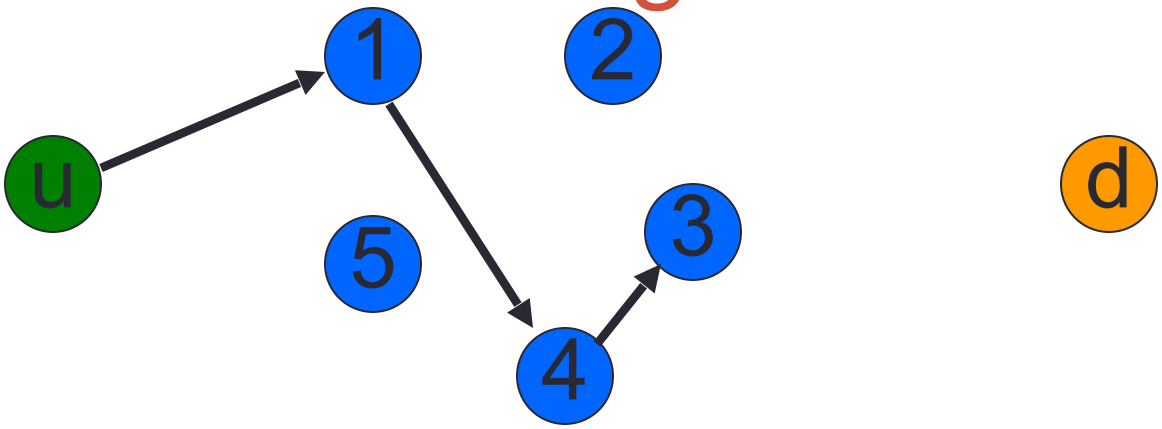
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



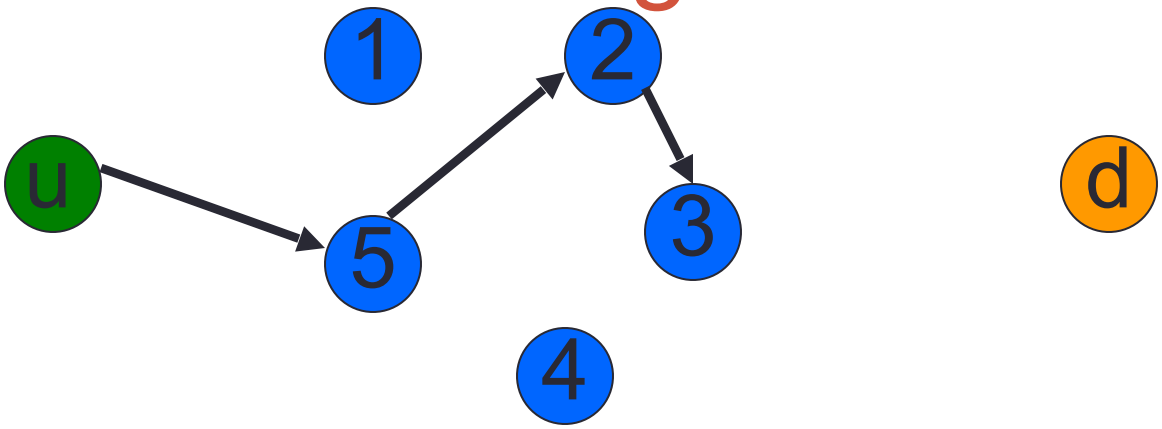
1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged

How Onion Routing Works



1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged.
4. Stream is closed.

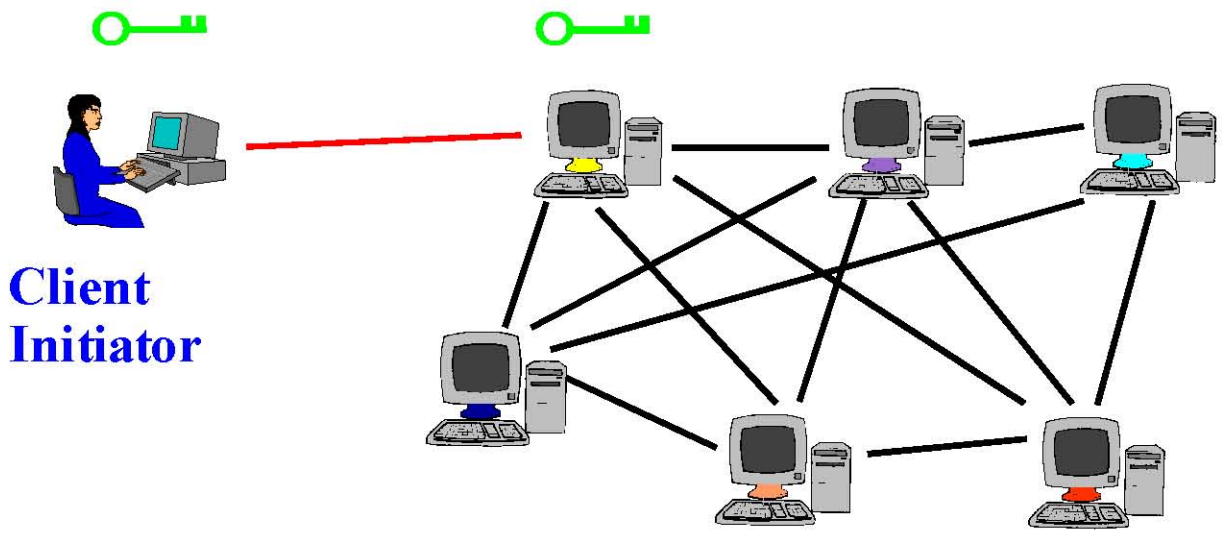
How Onion Routing Works



1. *u* creates *l*-hop circuit through routers
2. *u* opens a stream in the circuit to *d*
3. Data are exchanged.
4. Stream is closed.
5. Circuit is changed every few minutes.

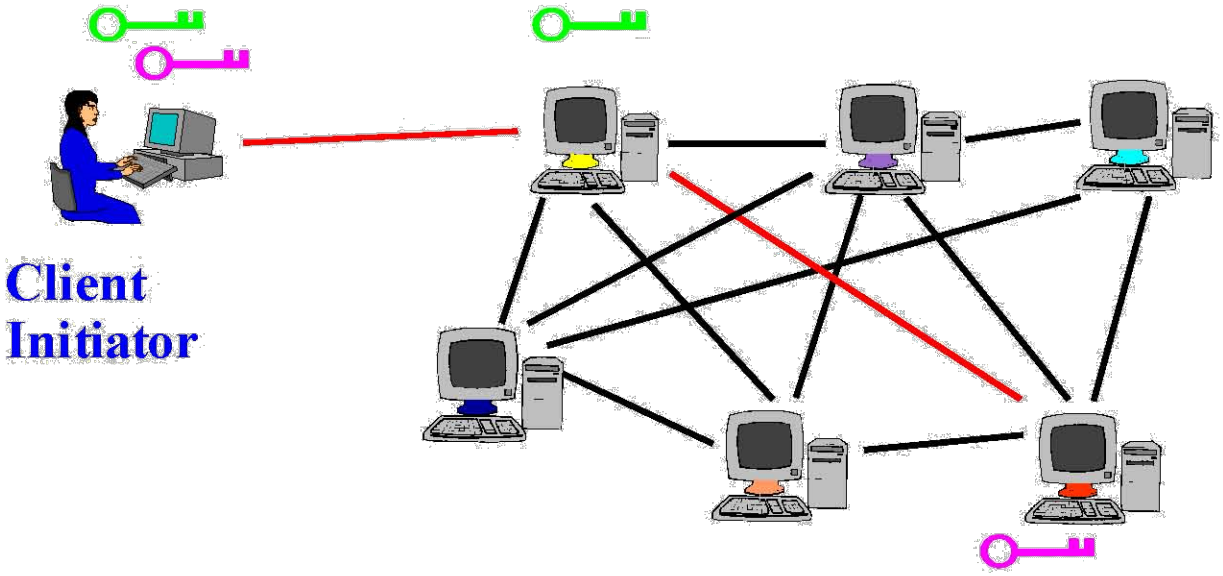
Tor Circuit Setup (1)

- Client proxy establish a symmetric session key and circuit with relay node #1



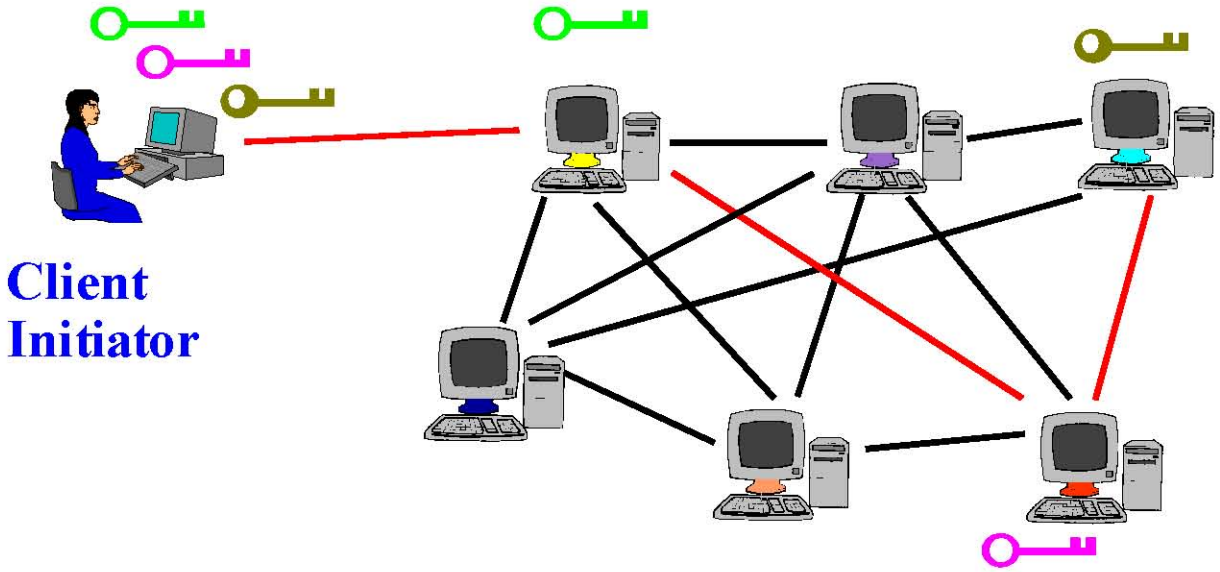
Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with relay node #2
  - Tunnel through relay node #1 - don't need !



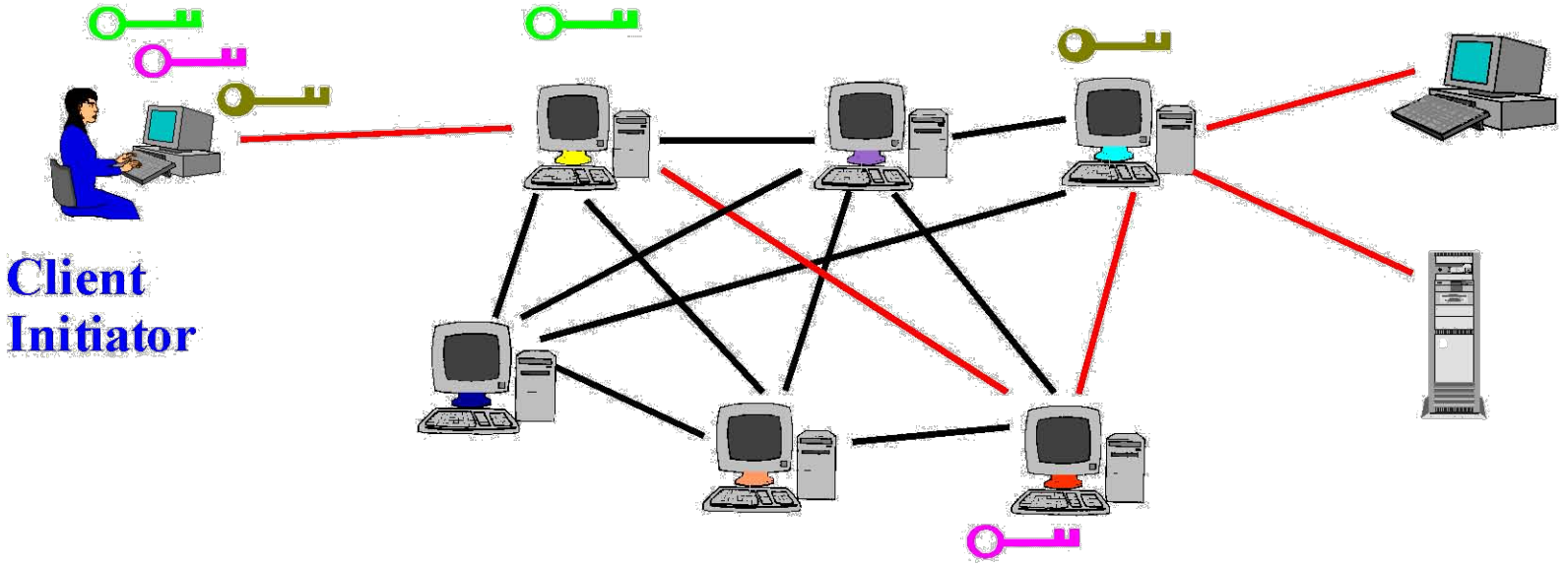
Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with relay node #3
  - Tunnel through relay nodes #1 and #2



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit
  - Datagrams decrypted and re-encrypted at each link



Additional functionality

- Integrity checking
  - Only done at the edges of a stream
  - SHA-1 digest of data sent and received
  - First 4 bytes of digest are sent with each message for verification
- OR-to-OR congestion might happen if too many users choose the same OR-to-OR connection.
- Circuit Level throttling
  - 2 windows keep track of relay data to be transmitted to other ORs (packaging window) and data transmitted out of the network (delivery window)
  - Windows are decremented after forwarding packets and increments on a relay sendme message towards OP with streamID zero.
  - When a window reaches 0, no messages are forwarded

Design

- Overlay network on the user level
- Onion Routers (OR) route traffic
- Onion Proxy (OP) fetches directories and creates virtual circuits on the network on behalf of users.
- Uses TCP with TLS
- All data is sent in fixed size (bytes) cells

Using Tor

- Many applications can share one circuit
  - Multiple TCP streams over one anonymous connection
- Tor router doesn't need root privileges
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- Directory servers
  - Maintain lists of active relay nodes, their locations, current public keys, etc.
  - Control how new nodes join the network
    - "Sybil attack": attacker creates a large number of relays
  - Directory servers' keys ship with Tor code

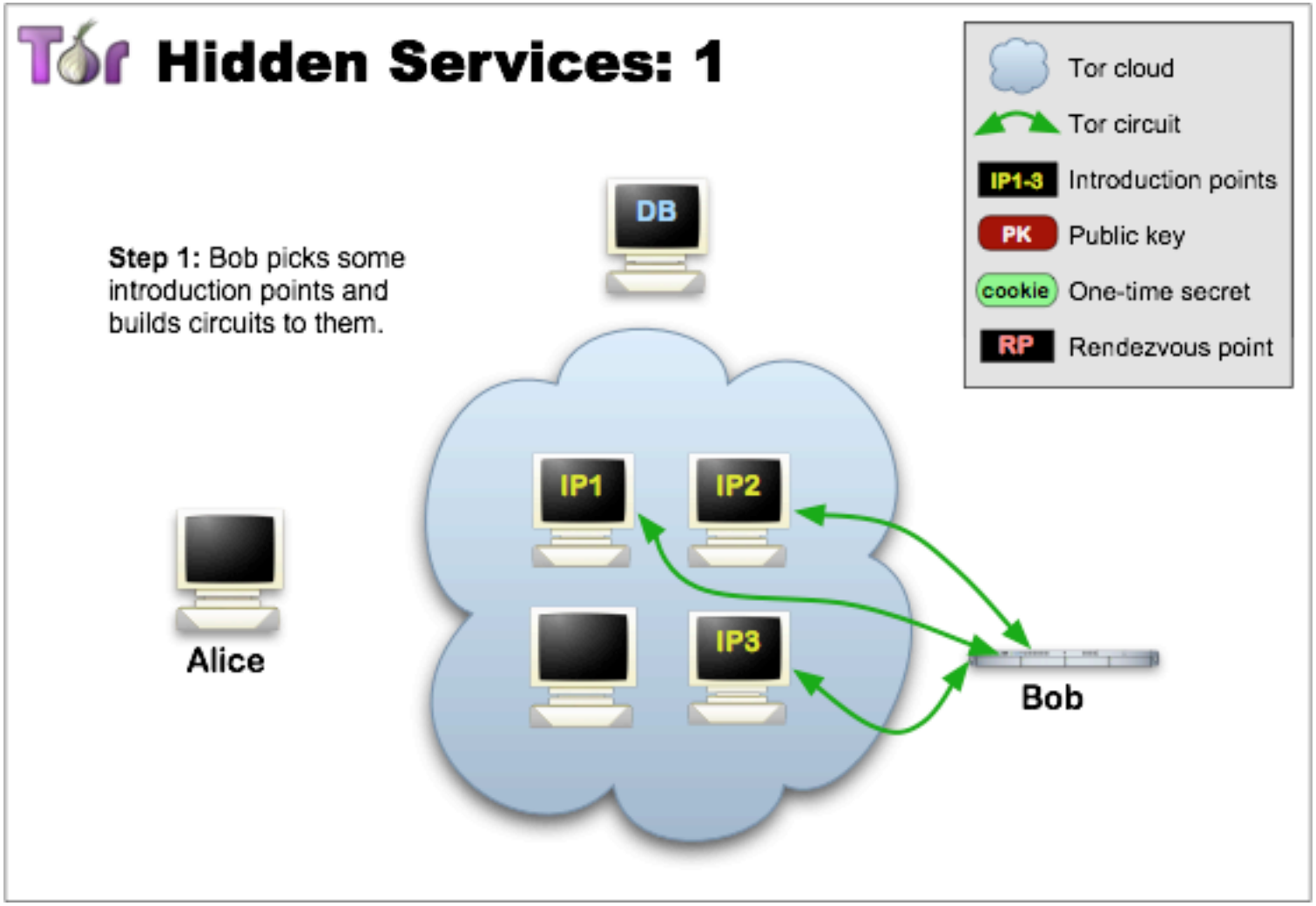
Hidden Services

- Goal: deploy a server on the Internet that anyone can connect to without knowing where it is or who runs it
- Accessible from anywhere
- Resistant to censorship, denial of service, physical attack
  - Network address of the server is hidden, thus can’t find the physical server

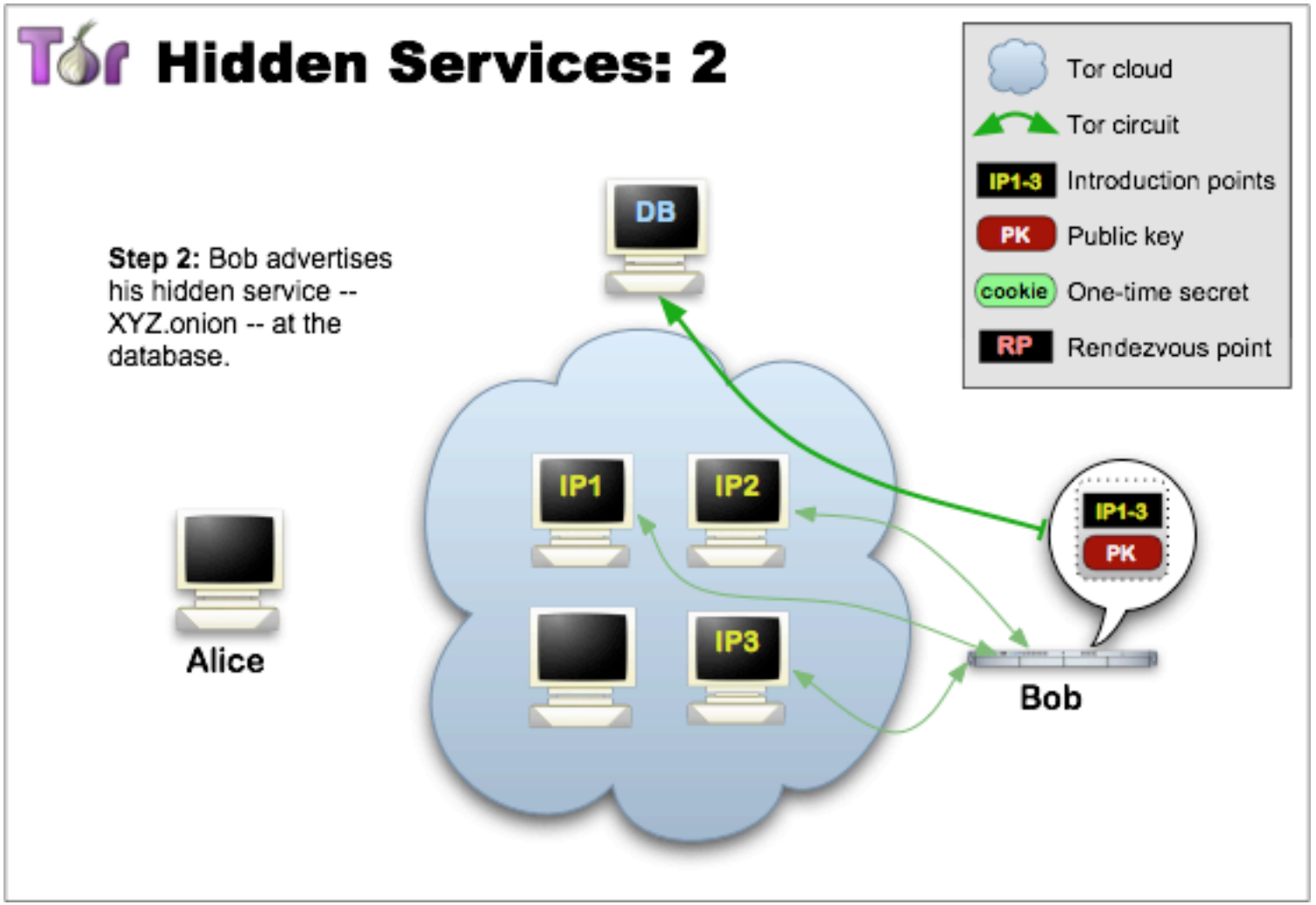
Hidden Service and Rendezvous Points

- Location-hidden services allow Bob to offer a TCP service without revealing his IP address.
- Tor accommodates receiver anonymity by allowing location hidden services
- Design goals for location hidden services
  - Access Control: filtering incoming requests (against flooding)
  - Robustness: maintain a long-term pseudonymous identity (ability to migrate service across Ors)
  - Smear-resistance: social attacker should not be able to “frame” a rendezvous router by offering an illegal location-hidden service and making observers believe the router created that service
  - Application transparency: same unmodified application
- Location hidden service leverage rendezvous points

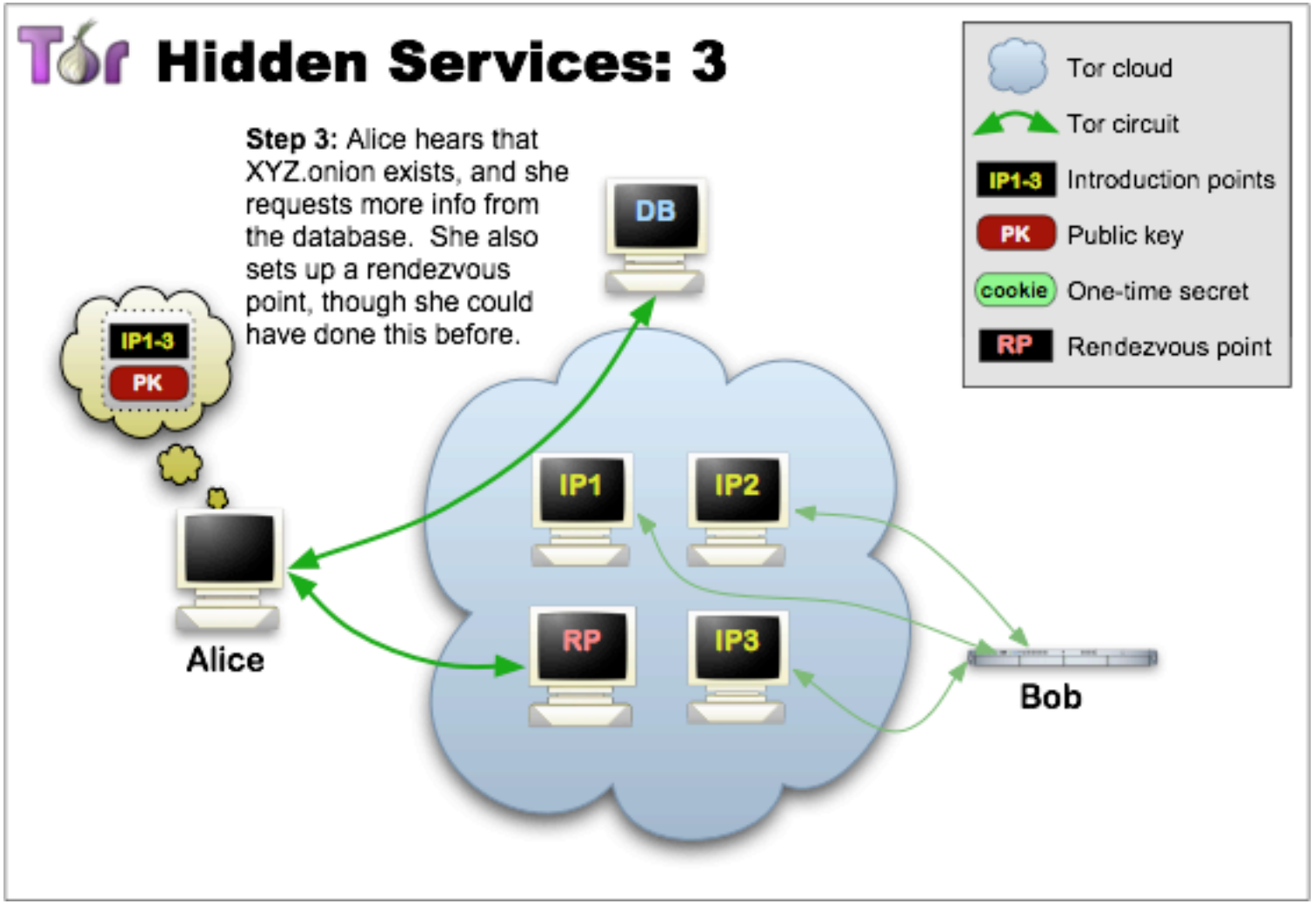
Setting up a hidden service 1



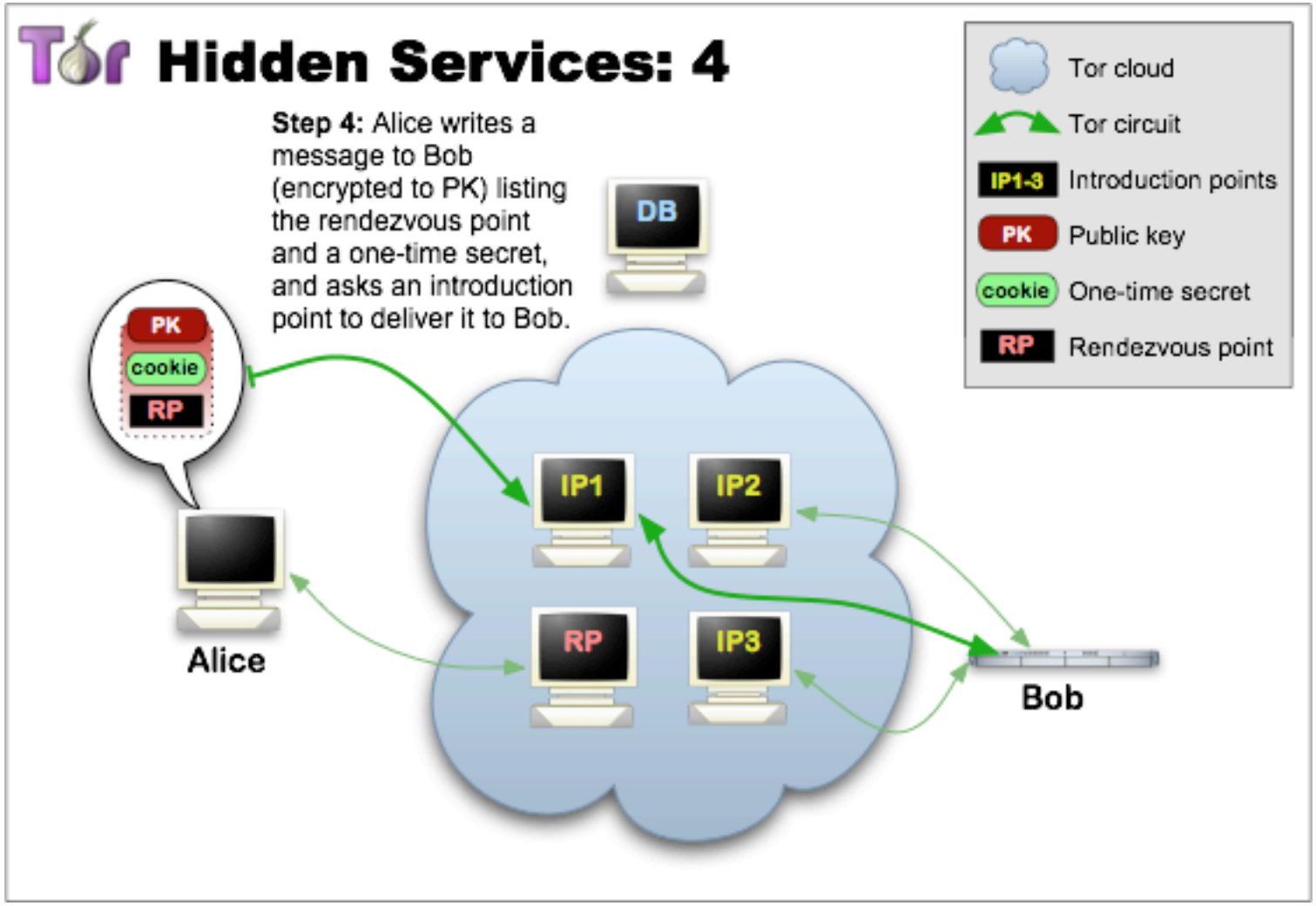
Setting up a hidden service 2



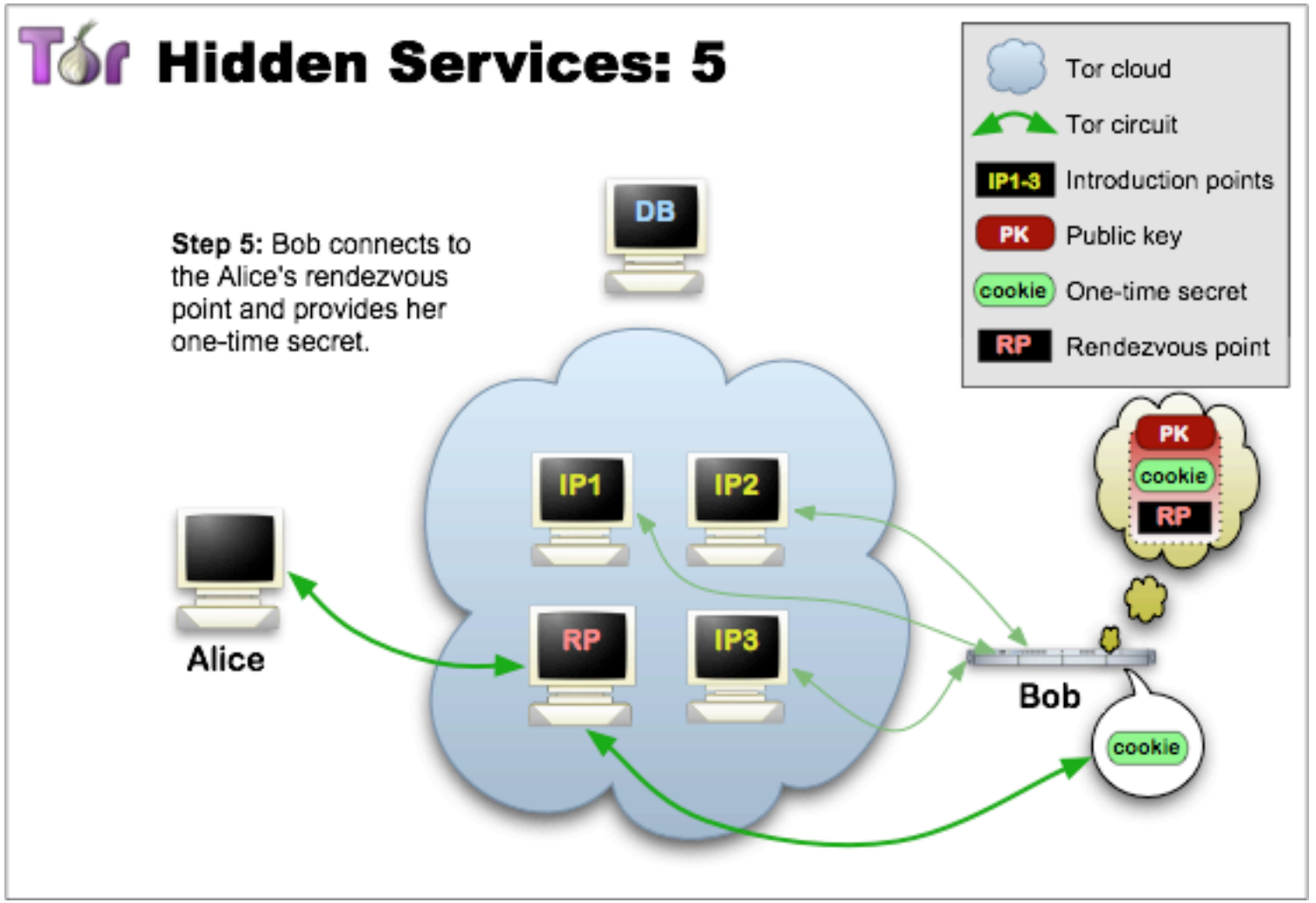
Setting up a hidden service 3



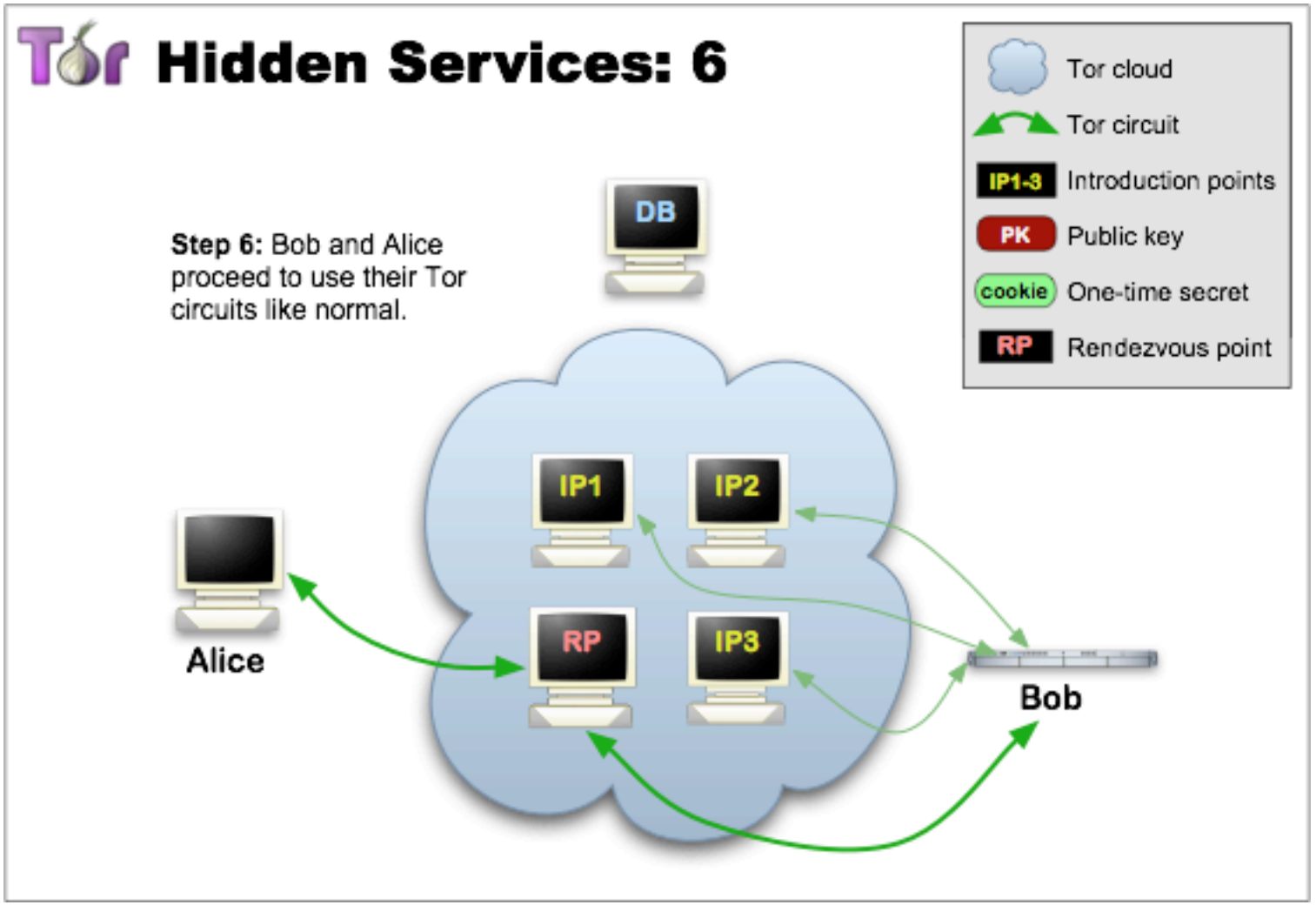
Setting up a hidden service 4



Setting up a hidden service 5



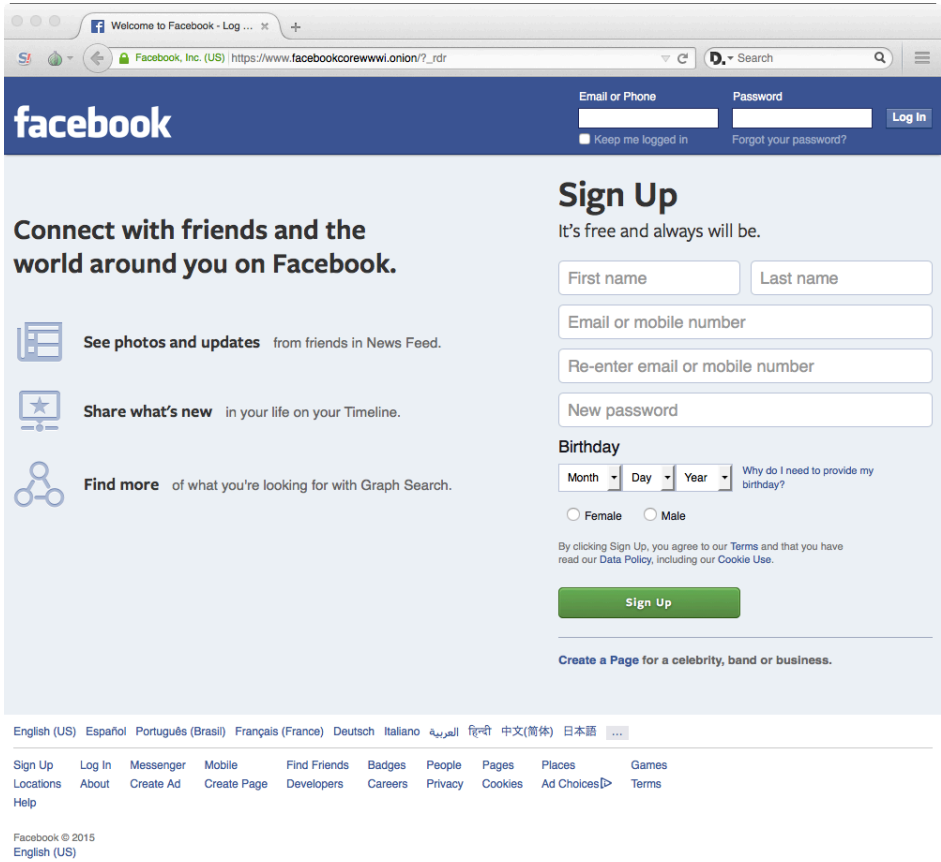
Setting up a hidden service 6



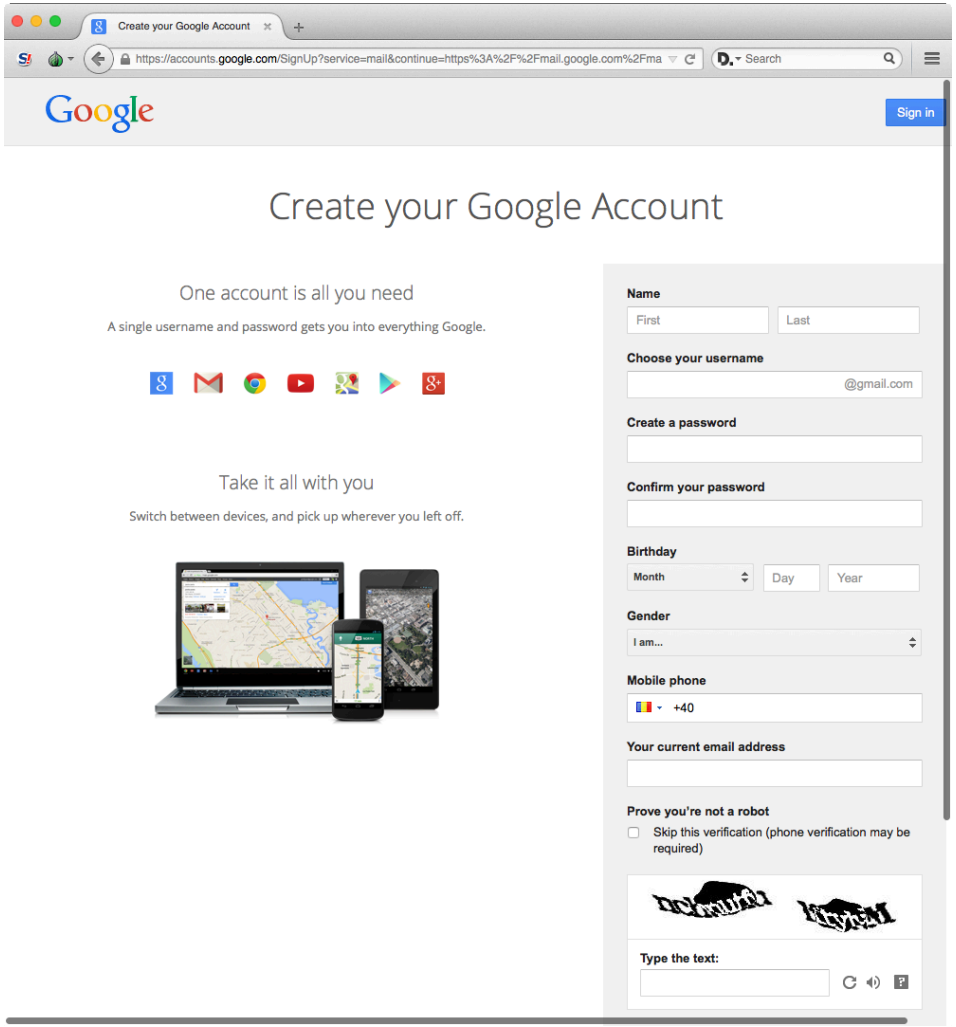
operations

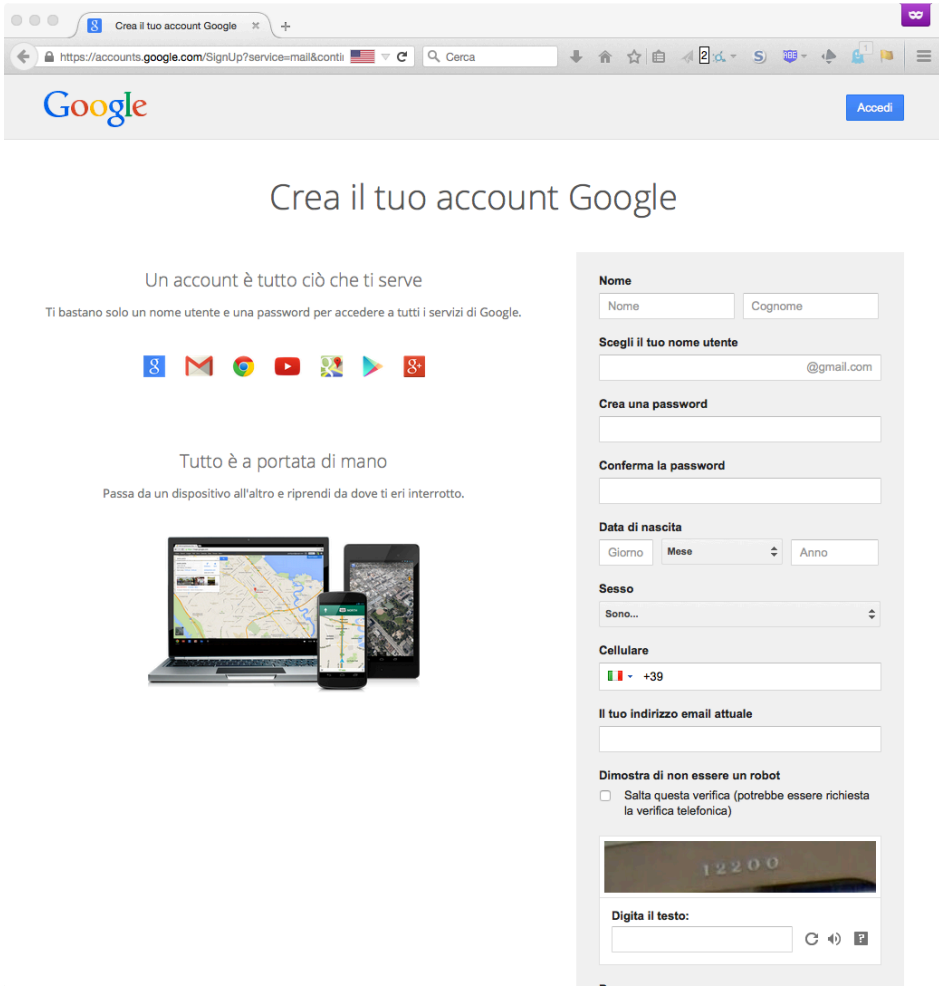
- Bob generates a long-term public key pair to identify his service
- Bob chooses some introduction points, and advertises them on the lookup service, also providing his public key
- Bob builds a circuit to each of his introduction points, and tells them to wait for requests.
- Alice learns about Bob's service out of band, retrieves the details of Bob's service from the lookup service
- Alice chooses an OR as the rendezvous point (RP) for her connection to Bob's service. She builds a circuit to the RP, and gives it a randomly chosen "rendezvous cookie" to recognize Bob.
- Alice opens an anonymous stream to one of Bob's introduction points, and gives it a message (encrypted with Bob's public key) telling it about herself, her RP and rendezvous cookie, and the start of a DH handshake. The introduction point sends the message to Bob.
- If Bob wants to talk to Alice, he builds a circuit to Alice's RP and sends the rendezvous cookie, the second half of the DH handshake, and a hash of the session key they now share.
- The RP connects the two circuits. Note that RP can't recognize Alice, Bob, or data they transmit.
- Alice sends a *relay begin* cell along the circuit. It arrives at Bob's OP, which connects to Bob's webserver.
- An anonymous stream has been established, and Alice and Bob communicate as normal.

hidden services examples

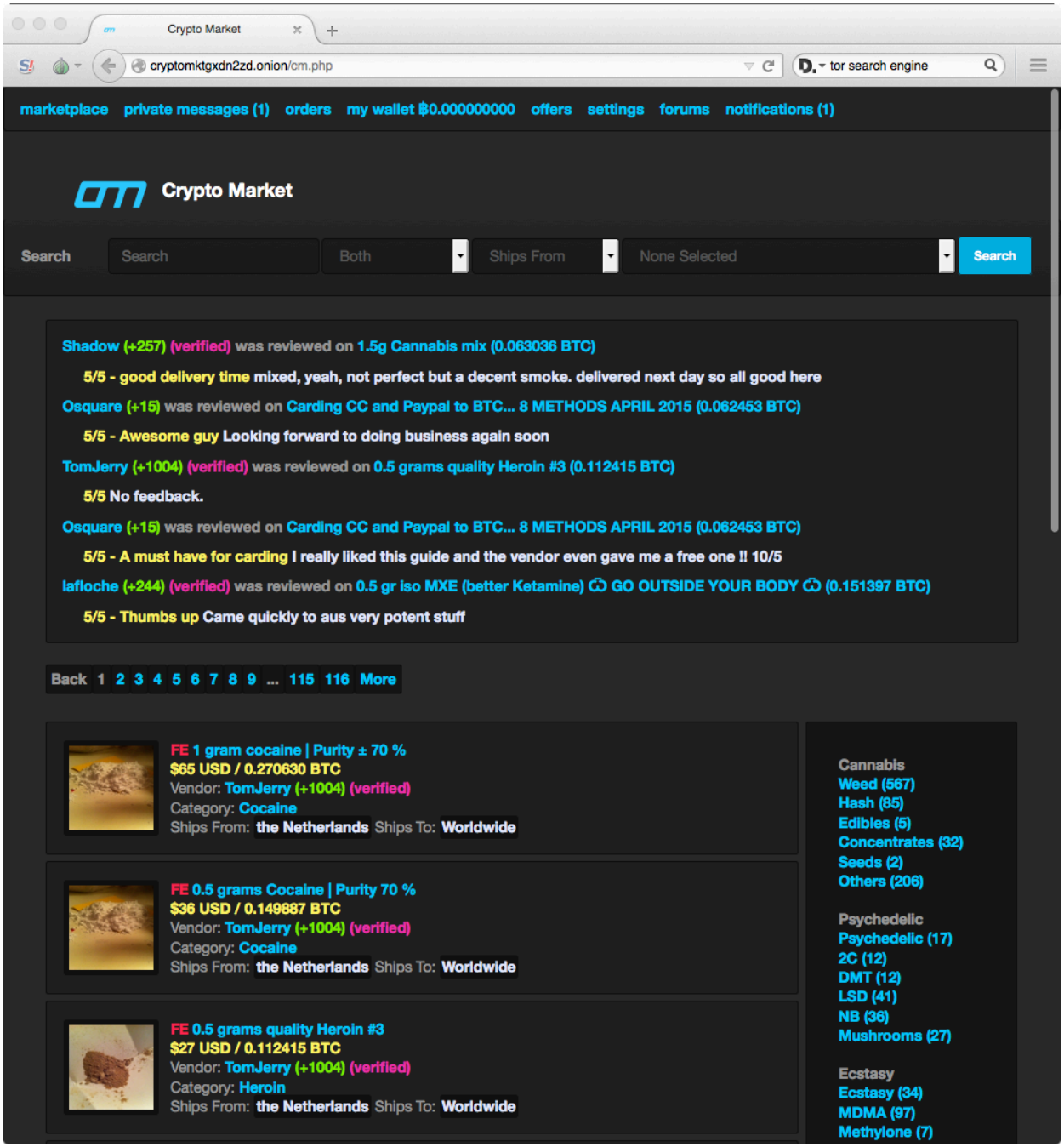


how Google treats Tor





crypto market



more insight

- https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf
- http://freehaven.net/anonbib/
- https://www.torproject.org/docs/hidden-services.html.en
- https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt