Tor

and onion routing

Onion Routing

- Developed by Michael G. Reed, Paul F. Syverson, and David M. Goldschlag, and patented by the United States Navy in US Patent No. 6266704 (1998)

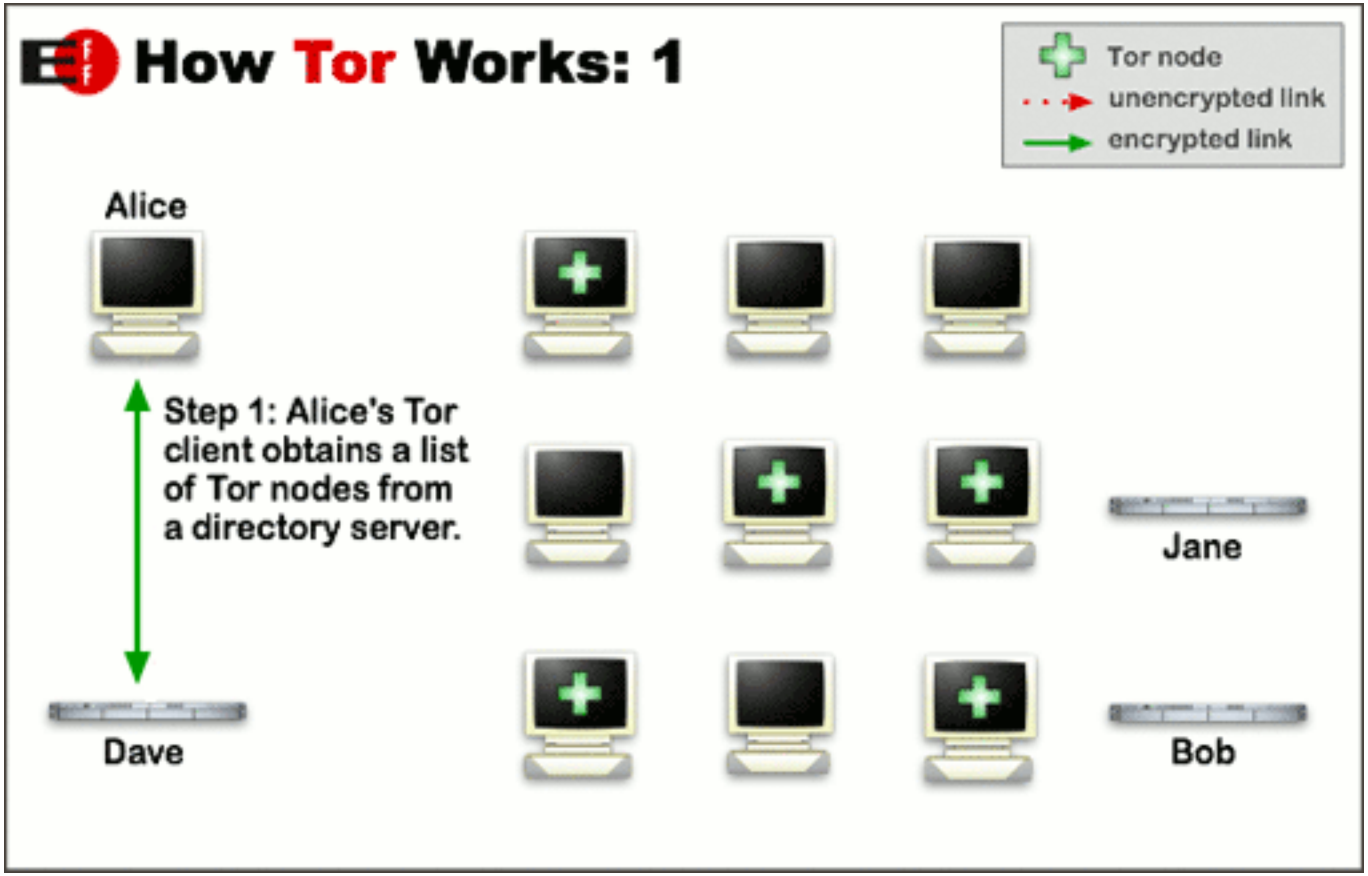
Tor

- On August 13, 2004 at the 13th USENIX Security Symposium, Roger Dingledine, Nick Mathewson, and Paul Syverson presented Tor, The Second-Generation Onion Router.
- Tor is unencumbered by the original onion routing patents, because it uses telescoping circuits. Tor provides perfect forward secrecy and moves protocol cleaning outside of the onion routing layer, making it a general purpose TCP transport. It also provides low latency, directory servers, end-to-end integrity checking and variable exit policies for routers. Reply onions have been replaced by a rendezvous system, allowing hidden services and websites.
- The .onion pseudo-top-level domain is used for addresses in the Tor network.
- The Tor source code is published under the BSD license. As of January 2014, there are about 5,000 publicly accessible onion routers.
- https://en.wikipedia.org/wiki/Onion_routing

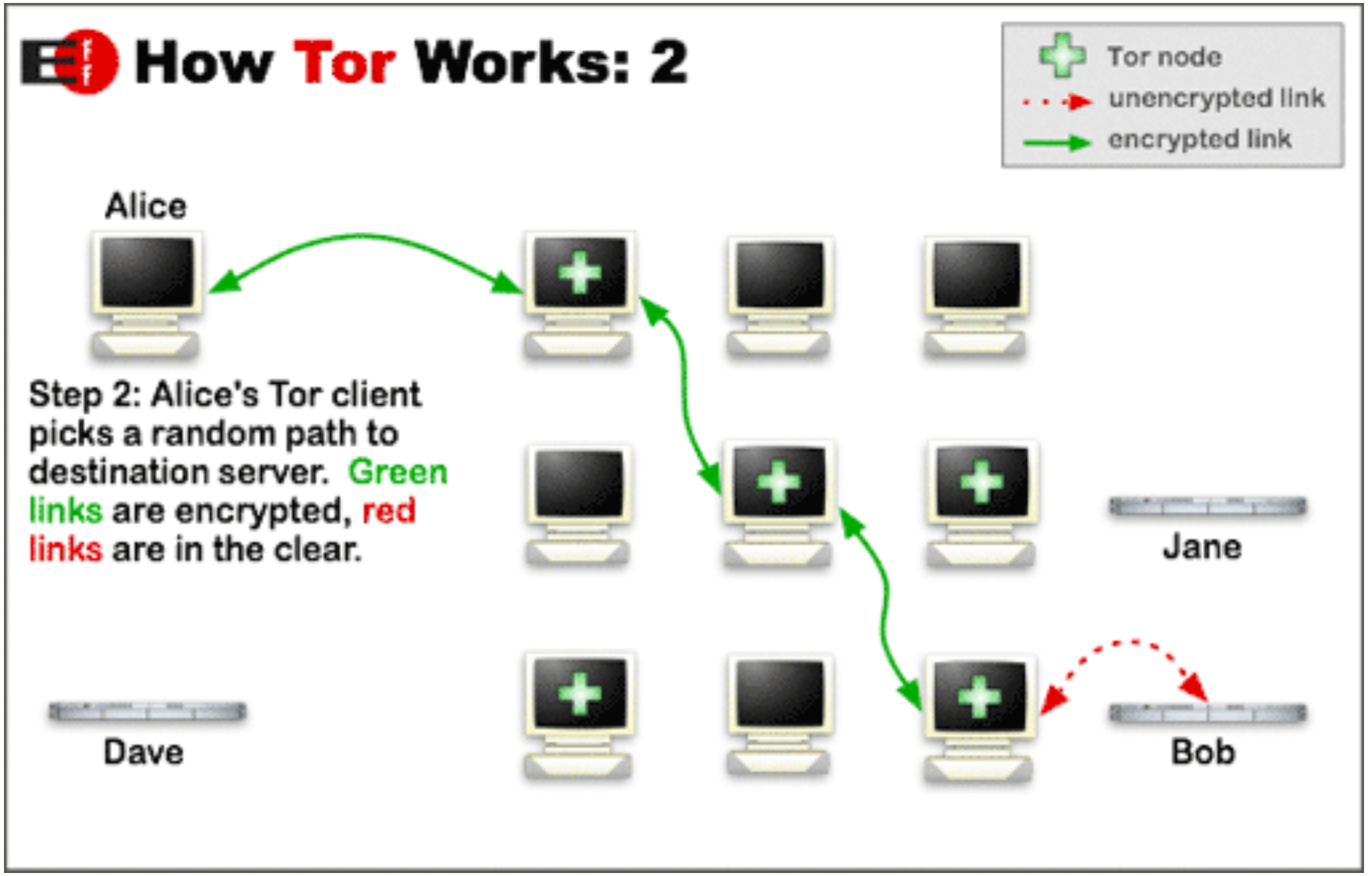
How Onion routing works

- Messages are repeatedly encrypted and then sent through several network nodes called onion routers
- Like someone peeling an onion, each onion router removes a layer of encryption to uncover routing instructions, and sends the message to the next router where this is repeated
- This prevents these intermediary nodes from knowing the origin, destination, and contents of the message

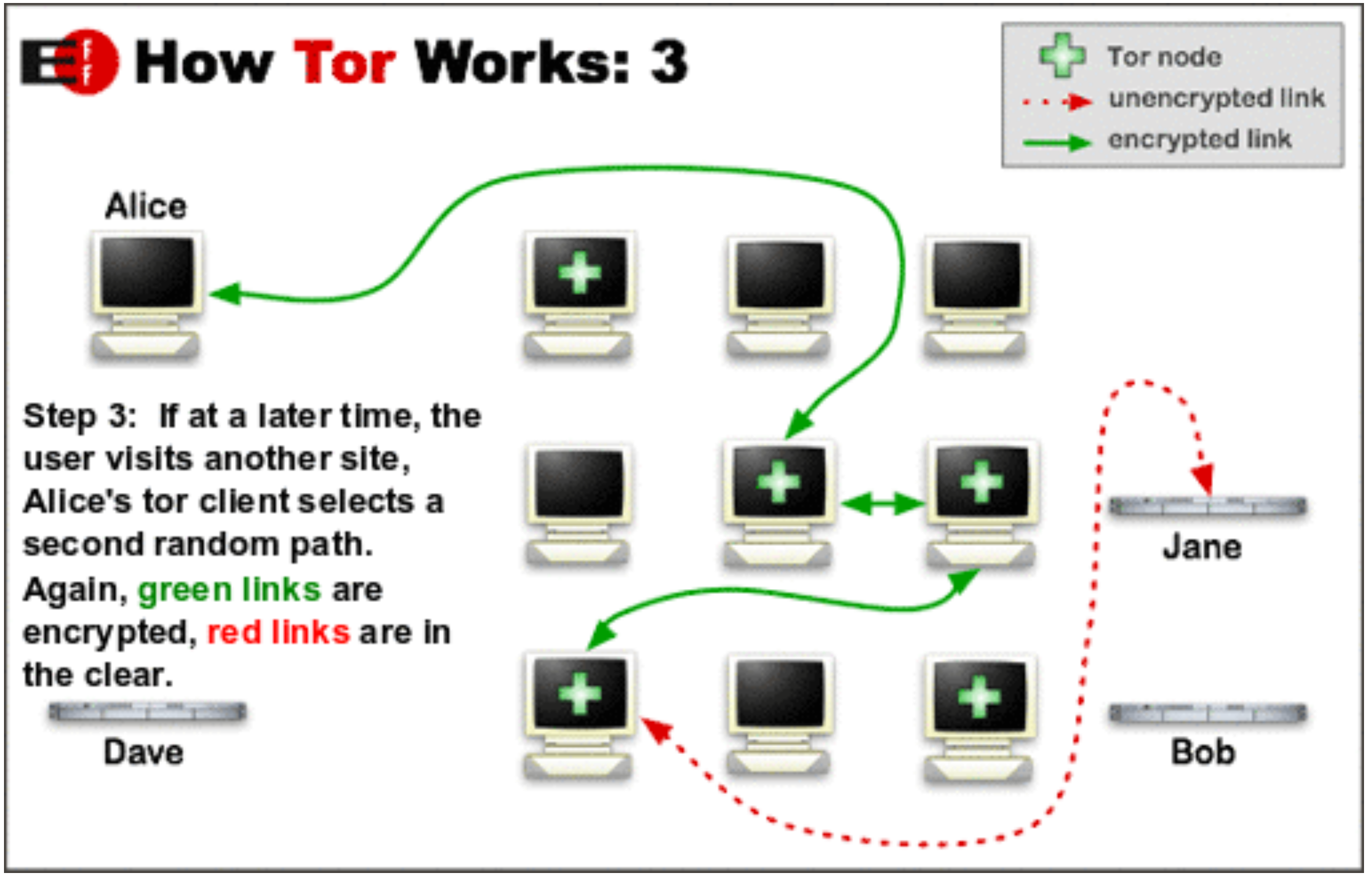
How Tor Works: 1



Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.

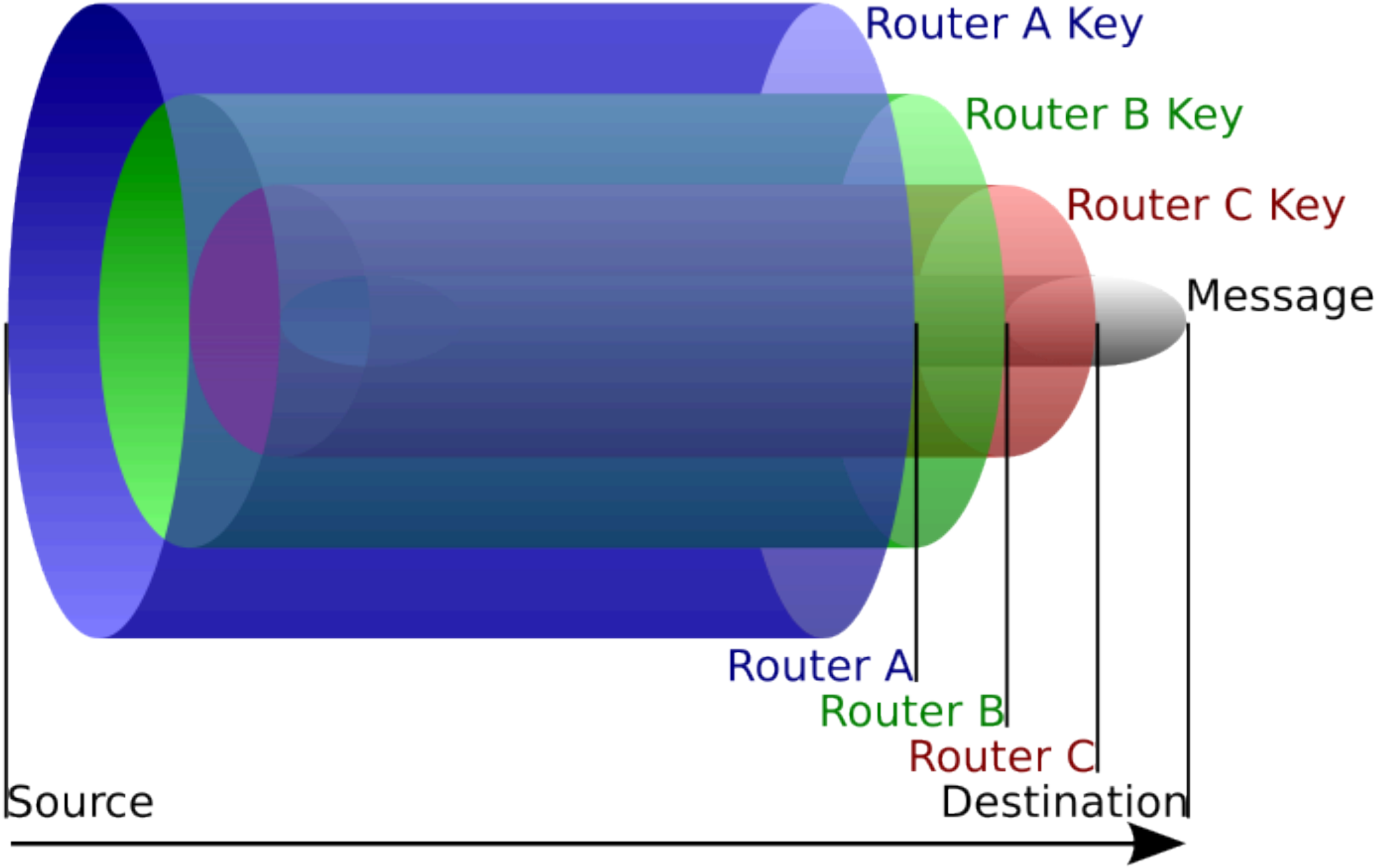


Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.



How Tor Works: 3

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.



Router A Key
Router B Key
Router C Key
Message
Router A
Router B
Router C
Source
Destination

Walkthrough (message out)

- Alice chooses a route to google.com through routers A, B, and C
- She gives the route an ID
- She then encrypts her message first with Router C’s public key, then Router B’s, then Router A’s.
- She then sends the message to router A who decrypts and sees the address for router B, stores the ID with the next/previous hop addresses, and sends it off
- Router B decrypts and sees the address for router A, stores the ID with the next/previous hop addresses, and sends it off
- Router C decrypts and sees the address for google.com, a session key for encryption of cleartext contents, stores the ID with the next/previous hop addresses, and sends it off

Walkthrough (message back)

- Router A (the exit node) has a session key shared with Alice and uses it to encrypt the message
- Encrypted message is sent back to Router B
- Router B knows that messages received from A with a particular ID should be sent back to Router C
- Router C knows that messages received from B with a particular ID should be sent back to Alice
- Alice then views the contents of the message

Notes

- The last router on the way out is the “exit node”
- The first router on the way in is the “entry node”
- Node in between are “relays”
- Each router only sees one hop back and the next hop forward
- None see the contents of the message except the Exit node

Weaknesses

- Can you think of any?

Weaknesses

- Timing Analysis
  - An adversary could determine whether a node is communicating with a web by correlating when messages are sent by a server and when messages are received by a node. Tor, and any other low latency network, is vulnerable to such an attack. A node can defeat this attack by sending dummy messages whenever it is not sending or receiving real messages. This counter-measure is not currently part of the Tor threat model as it is considered infeasible to protect against this type of attack.

- Exit Node Sniffing
  - An exit node (the last node in a chain) has complete access to the content being transmitted from the sender to the recipient; Dan Egerstad, a Swedish researcher, used such an attack to collect the passwords of over 100 email accounts related to foreign embassies. However, if the message is encrypted by SSL, the exit node cannot read the information, just as any encrypted link over the regular internet.

- Sniper Attack
  - Jensen et al., describe denial of service attack targeted at the TOR node software, together with defences against that attack and variants. The attack works using a colluding client and server, and filling the queues of the exit node until the node runs out of memory, and hence can serve no other (genuine) clients. By attacking a significant proportion of the exit nodes this way an attacker can degrade the network, and increase the chance of targets using nodes controlled by the attacker.

Hidden Services

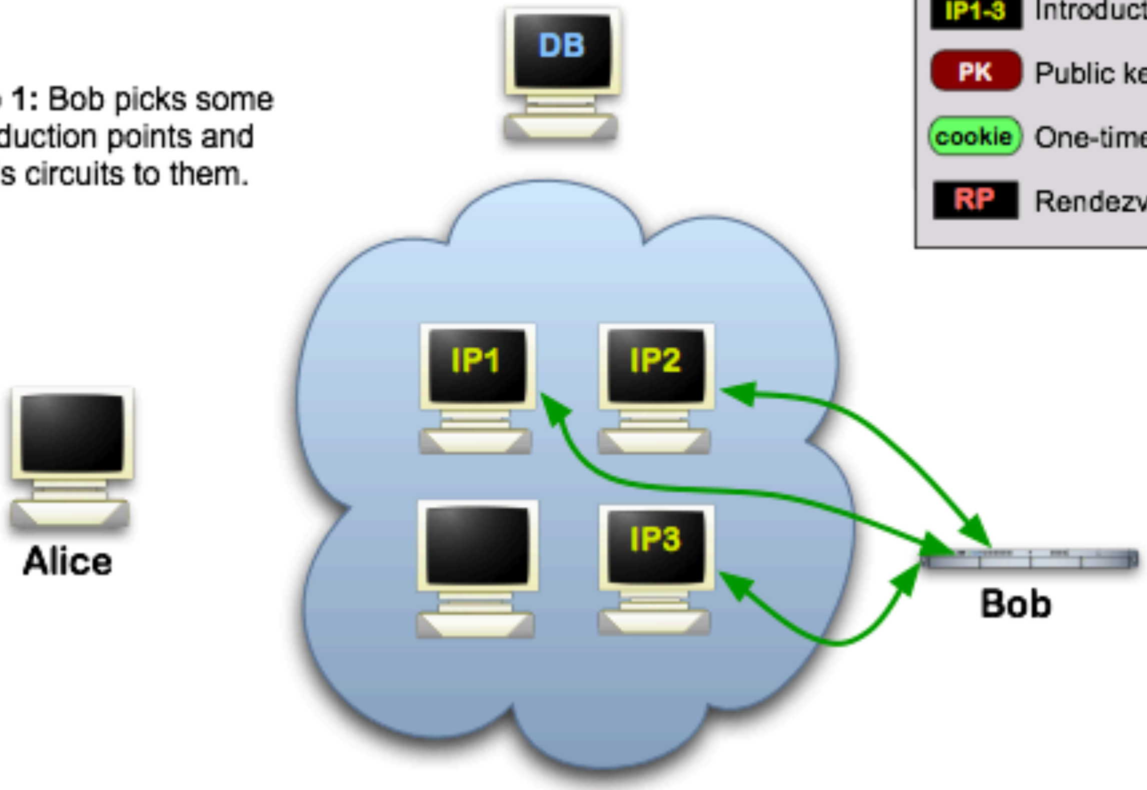
- No “exit nodes”
- Everything stays within the Tor network
- Tor’s traditional use hides only the client’s IP
- Hidden Services allow for both the client and host to engage in communication anonymously

Step 1

- A hidden service needs to advertise its existence in the Tor network before clients will be able to contact it. Therefore, the service randomly picks some relays, builds circuits to them, and asks them to act as introduction points by telling them its public key.

Tor Hidden Services: 1

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



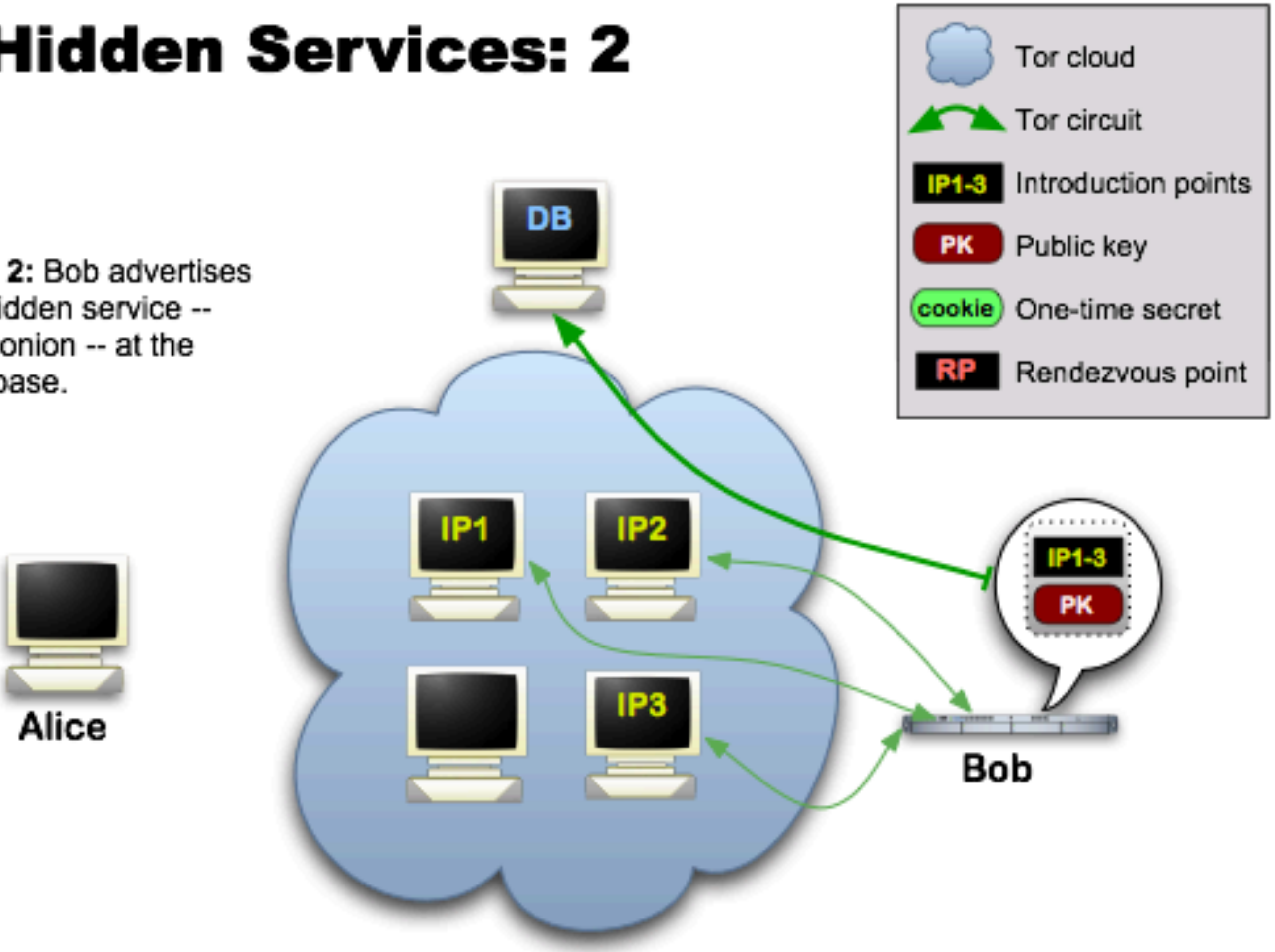
- Tor cloud
- Tor circuit
- IP1-3 Introduction points
- PK Public key
- cookie One-time secret
- RP Rendezvous point

Step 2

- The hidden service assembles a hidden service descriptor, containing its public key and a summary of each introduction point, and signs this descriptor with its private key.
- It uploads that descriptor to a distributed hash table.
- The descriptor will be found by clients requesting XYZ.onion where XYZ is a 16 character name derived from the service's public key.
- After this step, the hidden service is set up.

Tor Hidden Services: 2

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

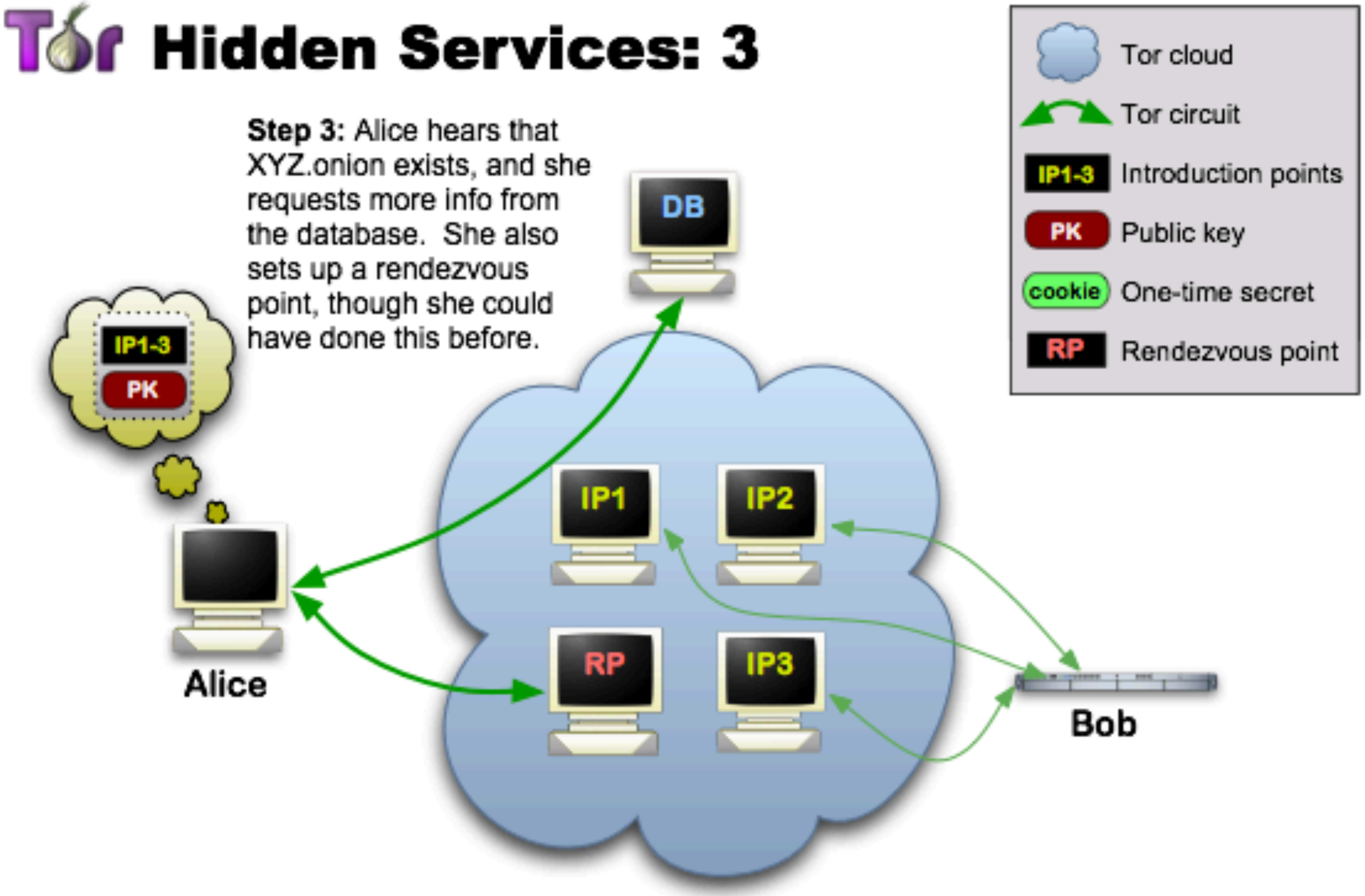


Step 3

- A client that wants to contact a hidden service needs to learn about its onion address first. After that, the client can initiate connection establishment by downloading the descriptor from the distributed hash table.
- If there is a descriptor for XYZ.onion (the hidden service could also be offline or have left long ago, or there could be a typo in the onion address), the client now knows the set of introduction points and the right public key to use.
- Around this time, the client also creates a circuit to another randomly picked relay and asks it to act as rendezvous point by telling it a one-time secret.

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.

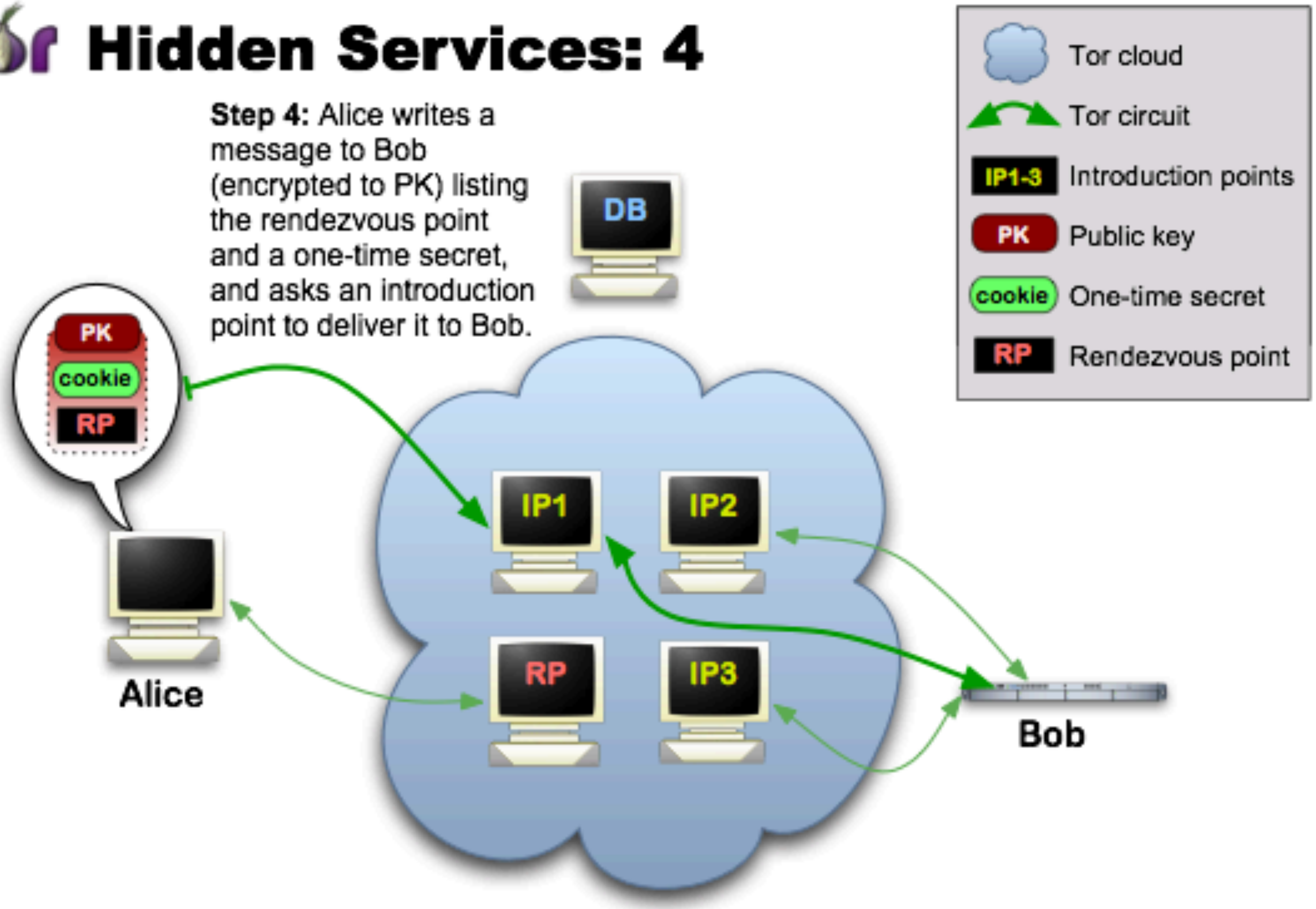


Step 4

- When the descriptor is present and the rendezvous point is ready, the client assembles an introduce message (encrypted to the hidden service's public key) including the address of the rendezvous point and the one-time secret.
- The client sends this message to one of the introduction points, requesting it be delivered to the hidden service.
- Again, communication takes place via a Tor circuit: nobody can relate sending the introduce message to the client's IP address, so the client remains anonymous.

Tor 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.

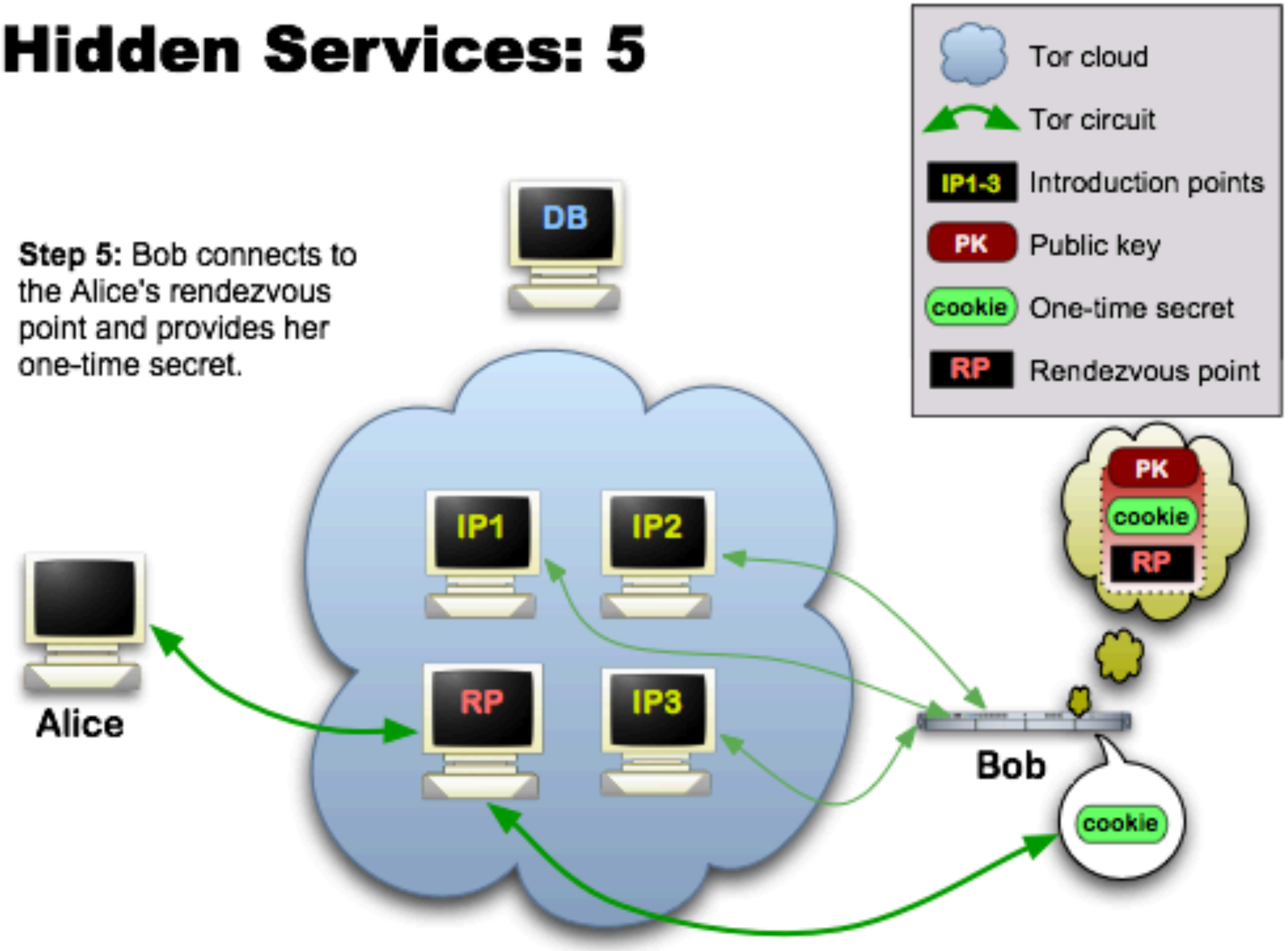


Step 5

- The hidden service decrypts the client's introduce message and finds the address of the rendezvous point and the one-time secret in it. The service creates a circuit to the rendezvous point and sends the one-time secret to it in a rendezvous message.

Tor Hidden Services: 5

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

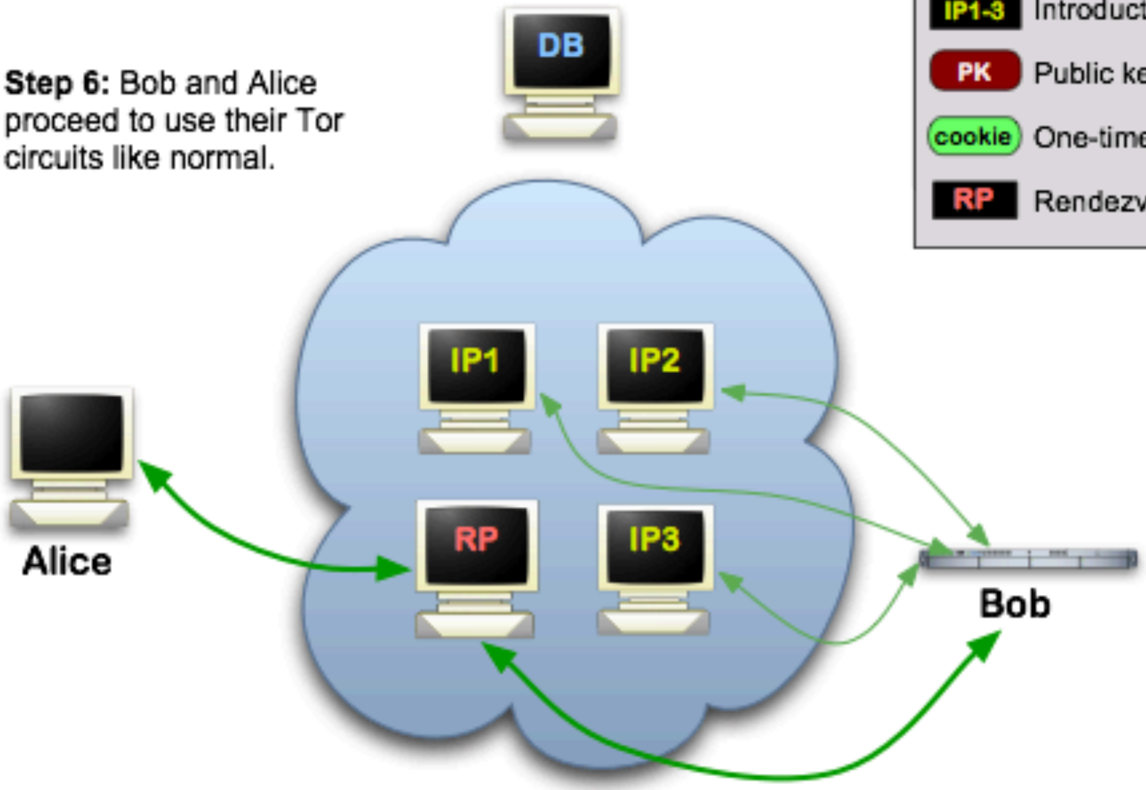


Step 6

- The rendezvous point notifies the client about successful connection establishment. After that, both client and hidden service can use their circuits to the rendezvous point for communicating with each other. The rendezvous point simply relays (end-to-end encrypted) messages from client to service and vice versa.

Tor Hidden Services: 6

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



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

Possible Attacks

- Can you think of any?