# TOR

By: Qui Hao (Frank) Yu and Haseeb Choudhary

# Agenda

What is Tor?

What does Tor provide its users?

The history of Tor

What makes up the Tor network?

How does Tor work?

- Clients perspective
- Onion service perspective

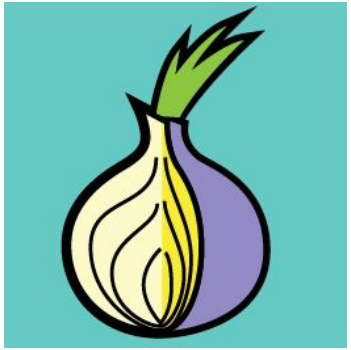
The Tor Browser

Tor with command line apps

Tor vs. VPN

What NOT to do with Tor

# What is Tor?



Tor stands for

**The Onion Router**

It is a open source network run by volunteers which provides its users with enhanced privacy and security on the Internet.

Guard Node

Middle Server

Exit Node

# What does Tor Provide its Users?

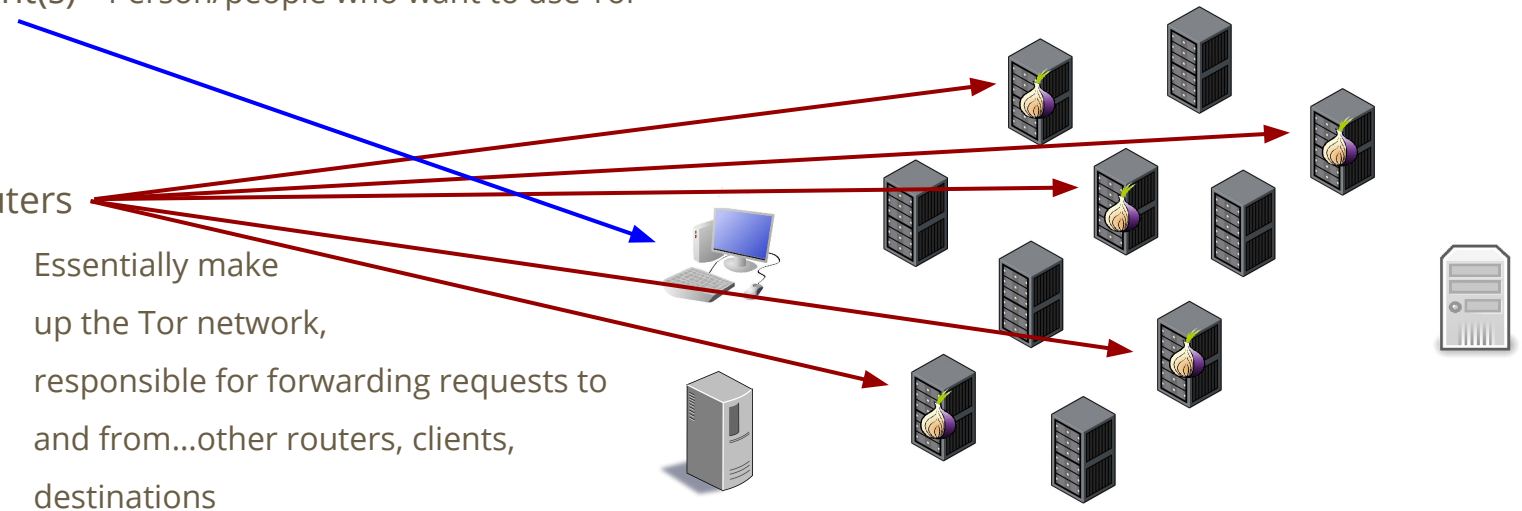
- Who are the users of Tor? Anyone who would like to enhance their privacy and security on the Internet
- What does Tor do to provide more privacy and security for its users?
- Tor prevents websites and other services from knowing your location
- Tor prevents someone monitoring your Internet traffic (e.g. ISP, someone on your home network) from learning where you're going and what you're receiving from where you go
- Tor routes your traffic through more than one Tor relay so that **no single relay will know both** who you are and where you're going

# The History of Tor

- Principle of “Onion Routing”
  - Developed by Paul Syverson, Michael G. Reed and David Goldschlag at the United States Naval Research Laboratory.
  - Developed in mid-1990’s
  - Purpose: protecting U.S. intelligence communication Online
- Alpha version of Tor - The Onion Routing Project
  - Developed by Roger Dingledine, Paul Syverson, and Nick Mathewson
  - Launched on Sept 20th, 2002
  - Releases a year later
- The Tor Project, Inc, founded in Dec, 2006

# What makes up the Tor Network?

- Client(s) - Person/people who want to use Tor
- Routers
  - Essentially make up the Tor network, responsible for forwarding requests to and from...other routers, clients, destinations
  - *Sort of* like routers in the Internet

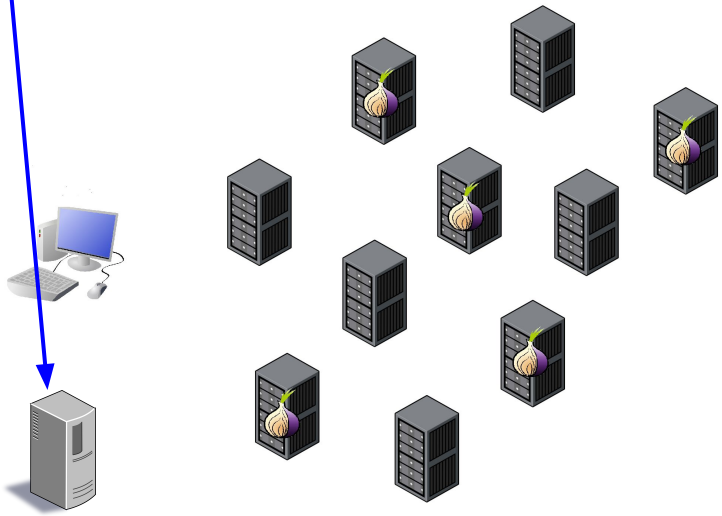


# Types of Onion Routers

- Entry nodes OR guard nodes
  - The first hop in the circuit
  - Knows who the client is and the next hop (middle node) in the circuit
- Middle node
  - The second hop in the circuit
  - Knows the first hop in the circuit and the next hop (exit node) in the circuit
- Exit node
  - The third and final hop in the circuit
  - Knows the second hop in the circuit and the destination

# What makes up the Tor Network? cont.

- Directory Authorities - Tell the client(s) which routers are available for use



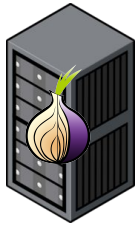
- Dark Web destinations [i.e. onion services] and clearnet destinations [e.g. wikipedia.org]
  - Onion services hosted anonymously in the Tor network - only accessible through Tor
  - Clearnet destinations not hosted in the Tor network - accessible with and without Tor

# Walkthrough of How Tor works

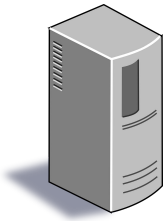
- Destination server

- Tor user

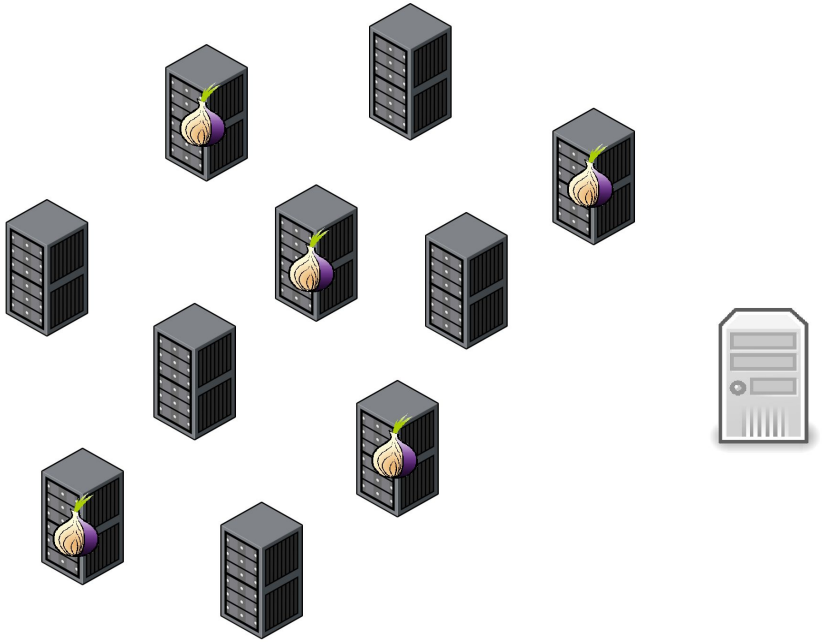
- Non-onion node/server



- Onion router

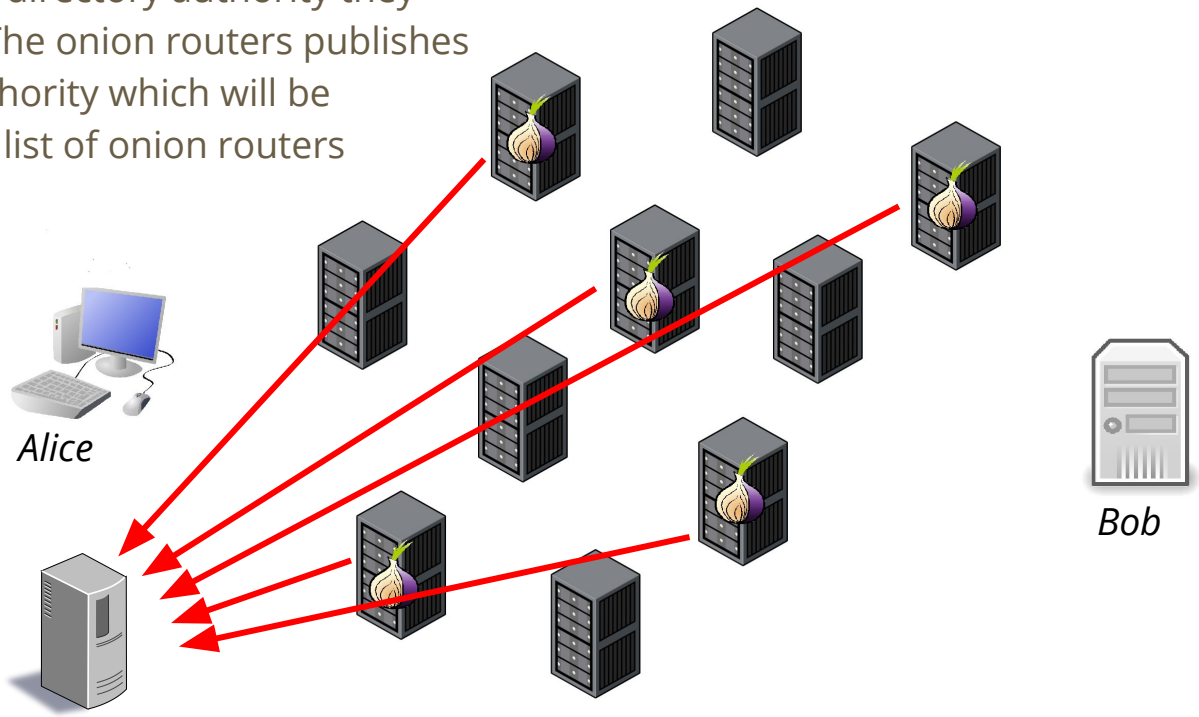


- Directory authority



# How Tor works

**Step 0a)** Onions routers tell the directory authority they are available as onion routers. The onion routers publishes information to the directory authority which will be given to a client who requests a list of onion routers from it.

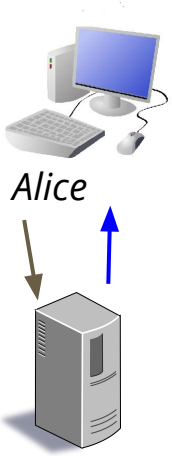


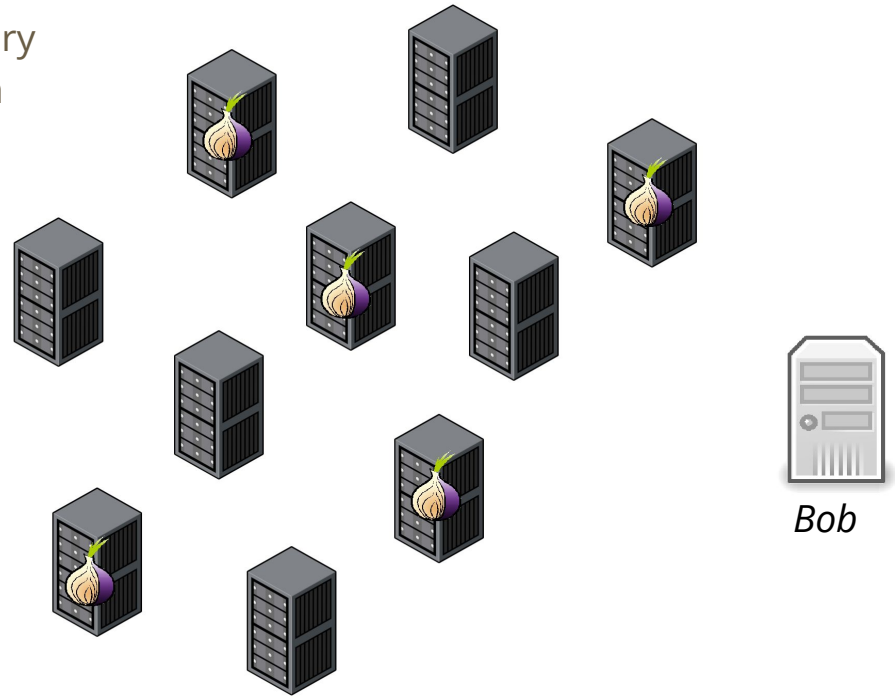
# How Tor works cont.

**Step 0b)** User Alice opens Tor Browser which contains a list of directory authorities. Alice requests the directory authority provide her with a list of available onion routers.

Alice: Hey, what onion routers are available?

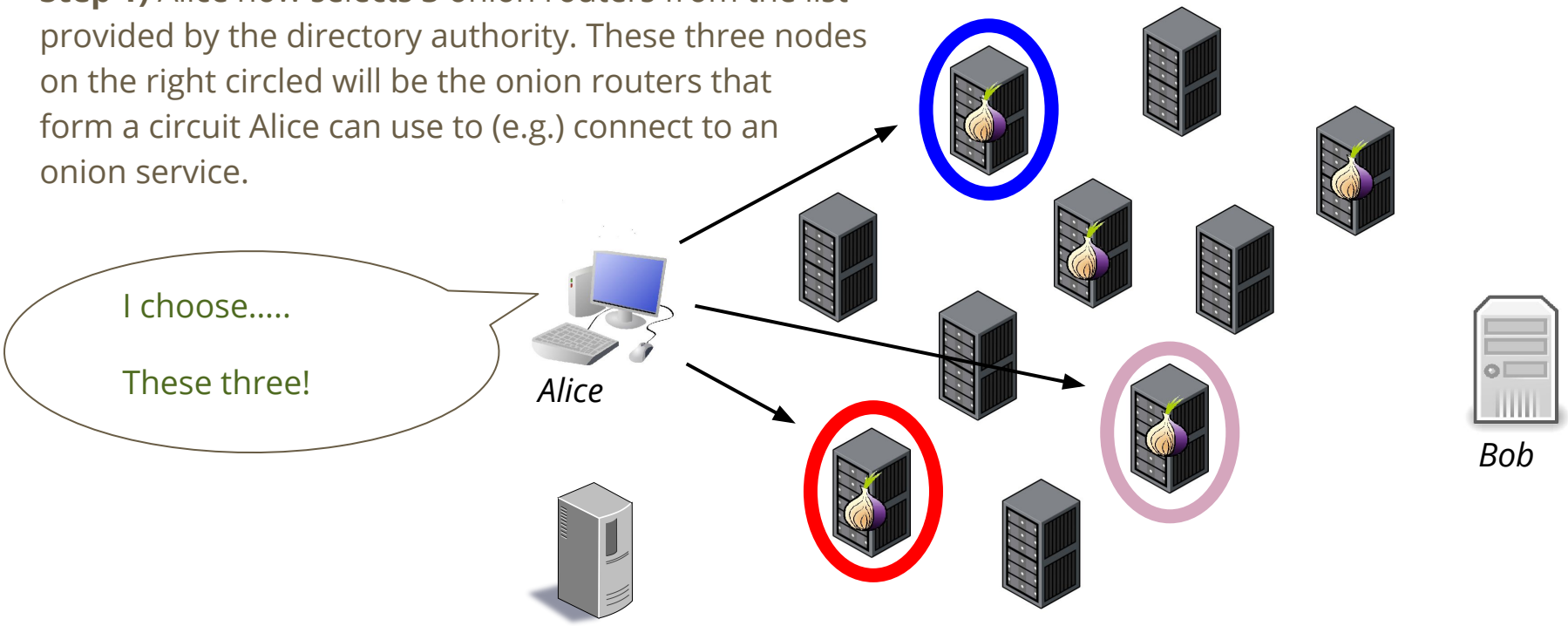
DA: Hey! Here's a list of them!



# How Tor works cont.

**Step 1)** Alice now selects 3 onion routers from the list provided by the directory authority. These three nodes on the right circled will be the onion routers that form a circuit Alice can use to (e.g.) connect to an onion service.



# A Few Questions...

- How do we form the circuit to connect and talk to a service?
- How do we stop a onion router from reading our message to the service?
- How do we stop adversaries from reading our message to the service?
- How do we set up encryption keys with the onion routers?

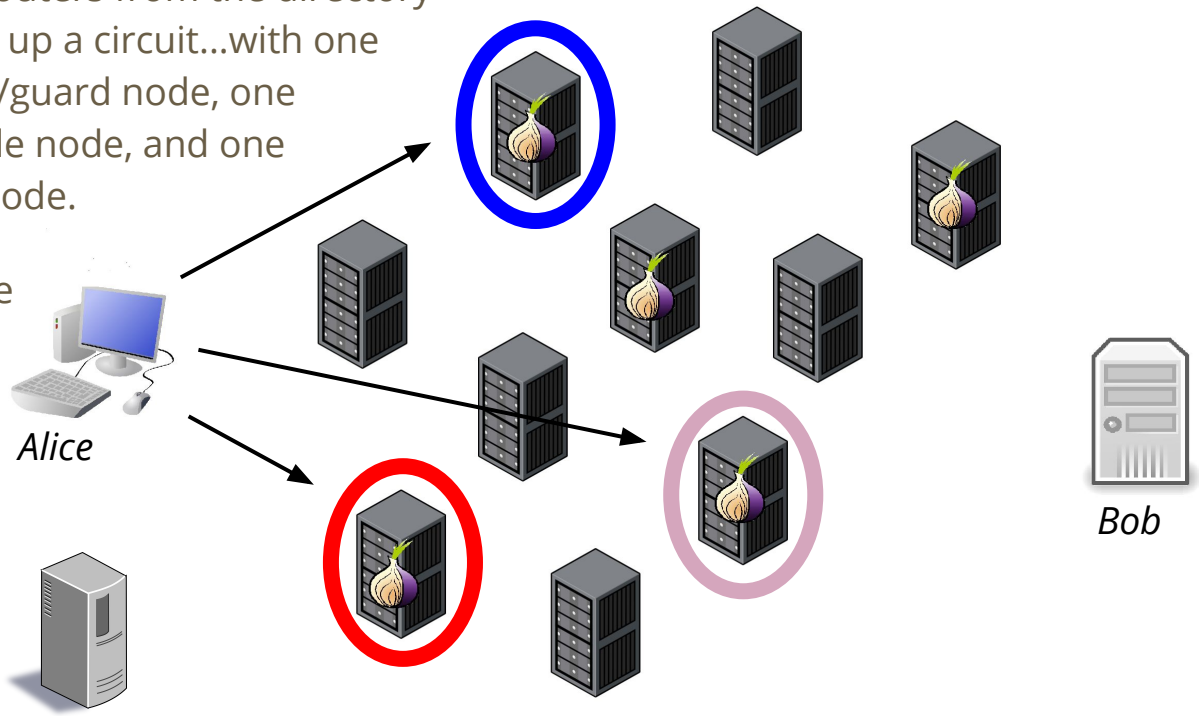
# Recall ...

- We can use the Diffie-Hellman key exchange for two entities to agree on a symmetric key used for encryption and decryption
- Diffie-Hellman on it's own is not **private** so the entities need a way to privately and securely communicate and perform the Diffie-Hellman key exchange
- One way: use asymmetric key encryption(/decryption) i.e. Public/Private key encryption/decryption
- Symmetric key encryption: AES

# Creating A Secure Circuit in the Tor Network

Alice knows these three onion routers from the directory authority. Now she needs to set up a circuit...with one onion router acting as the entry/guard node, one onion router acting as the middle node, and one onion router acting as the exit node.

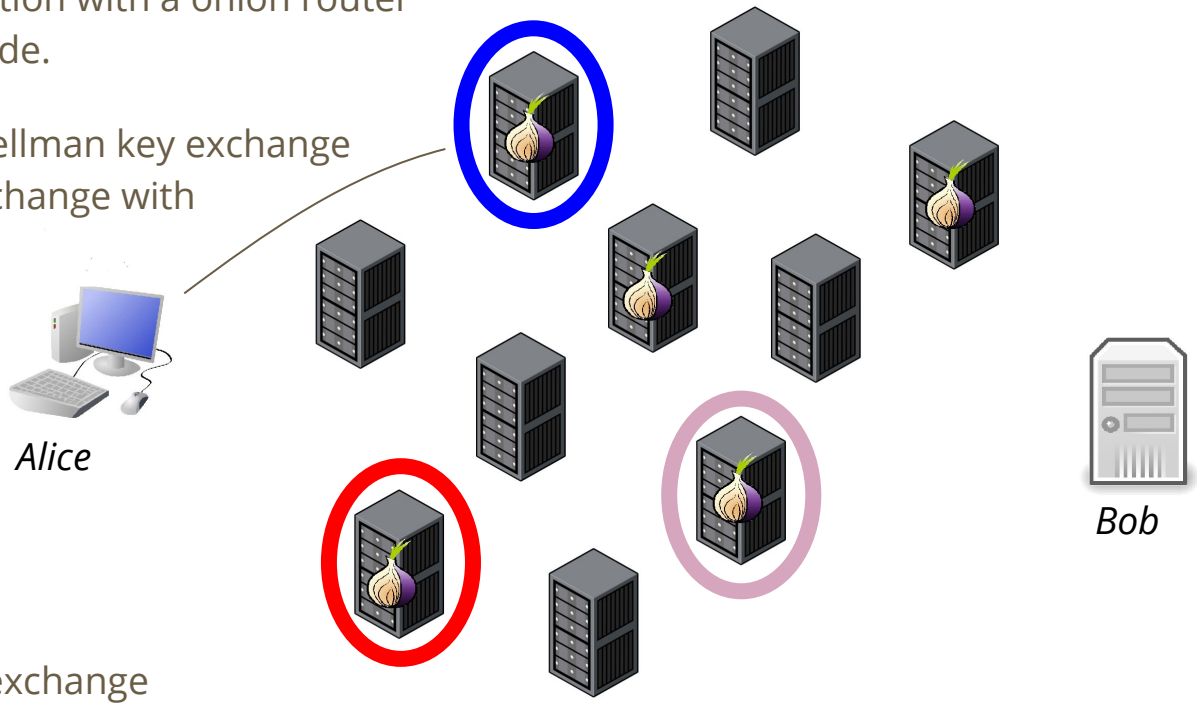
Alice can now request one of the onion routers to act as her entry node.



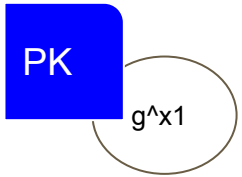
# Creating A Secure Circuit in the Tor Network cont.

Alice forms a secure TLS connection with a onion router requesting it act as her entry node.

Alice then starts off the Diffie-Hellman key exchange by encrypting her half of the exchange with the onion routers public key.



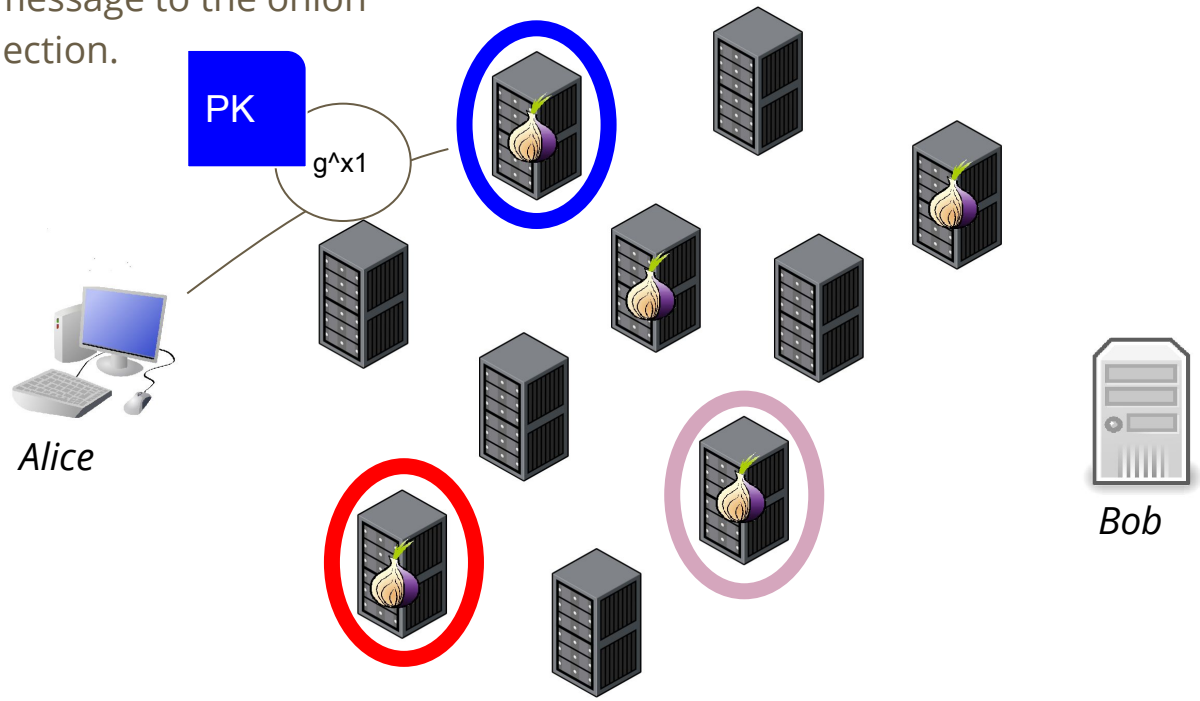

Onion routers public key




First half of DH key exchange

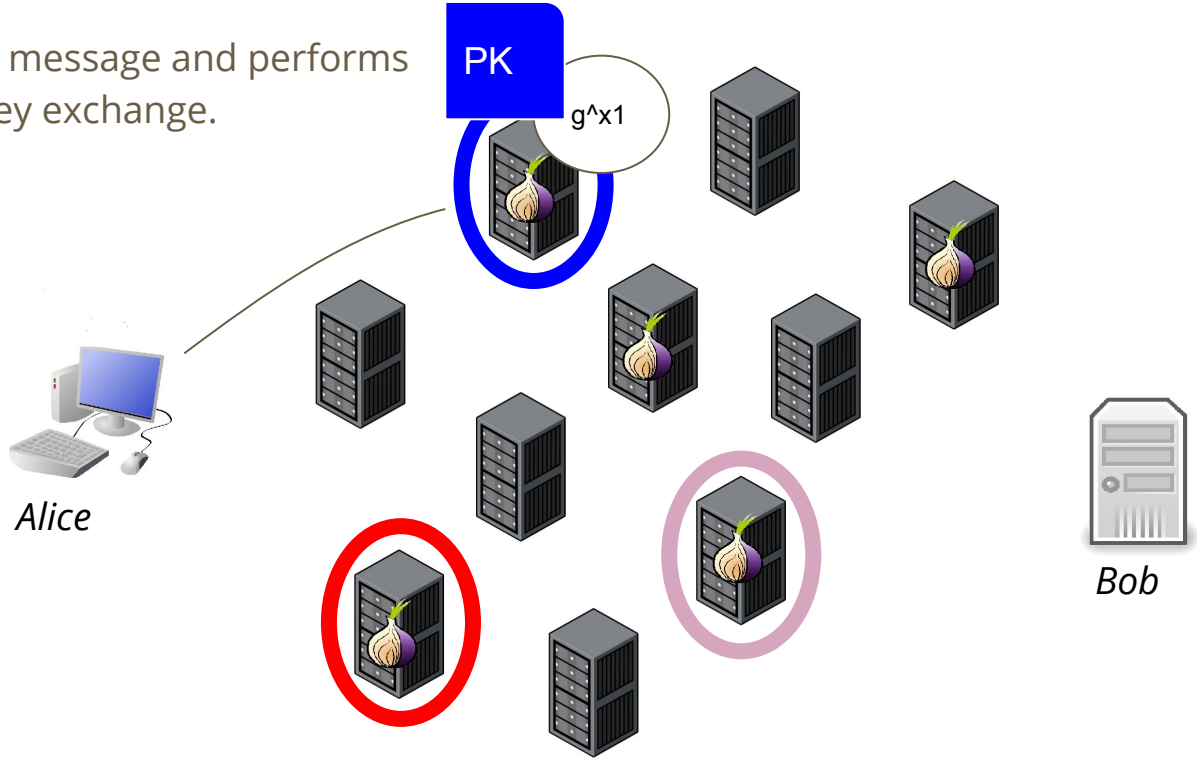
# Creating A Secure Circuit in the Tor Network cont.

Alice then sends the encrypted message to the onion router over the secure TLS connection.



# Creating A Secure Circuit in the Tor Network cont.

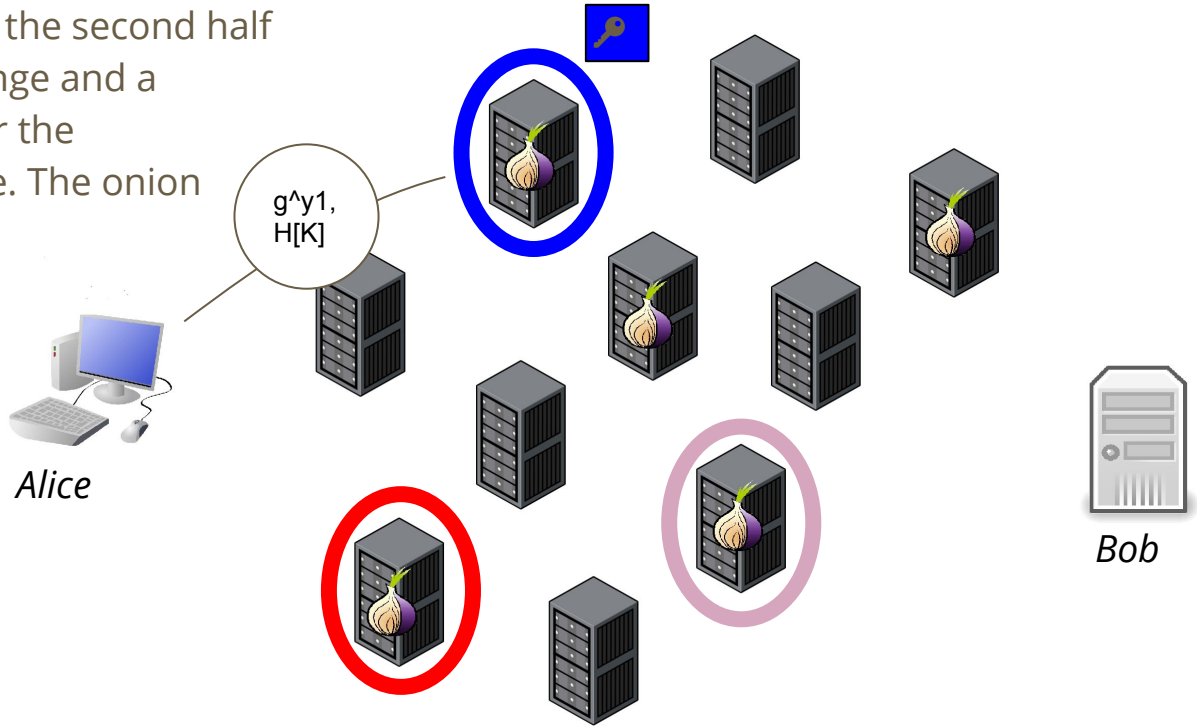
The onion router receives Alices message and performs its share of the Diffie-Hellman key exchange.



# Creating A Secure Circuit in the Tor Network cont.

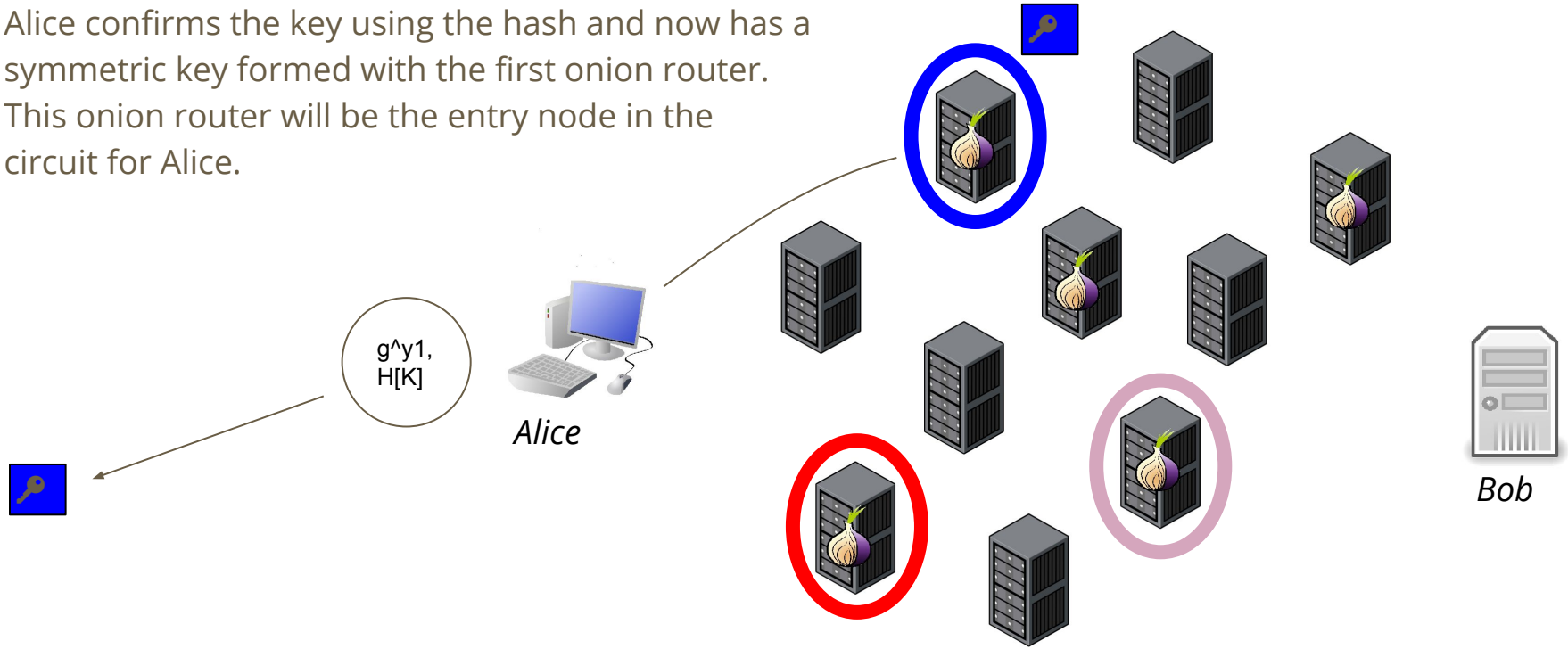
The onion router responds with the second half of the Diffie-Hellman key exchange and a hash of the key. This is sent over the secure TLS connection with Alice. The onion router knows the key that it will share with Alice.

H[K] - Hash of key



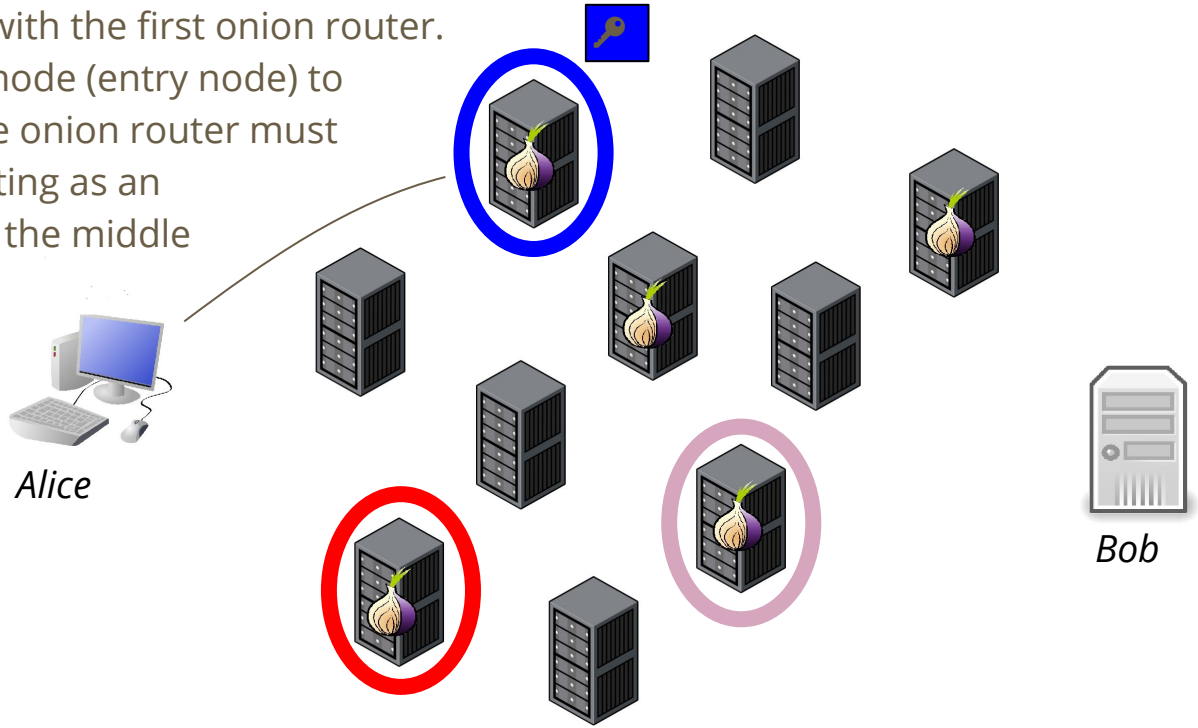
# Creating A Secure Circuit in the Tor Network cont.

Alice confirms the key using the hash and now has a symmetric key formed with the first onion router. This onion router will be the entry node in the circuit for Alice.



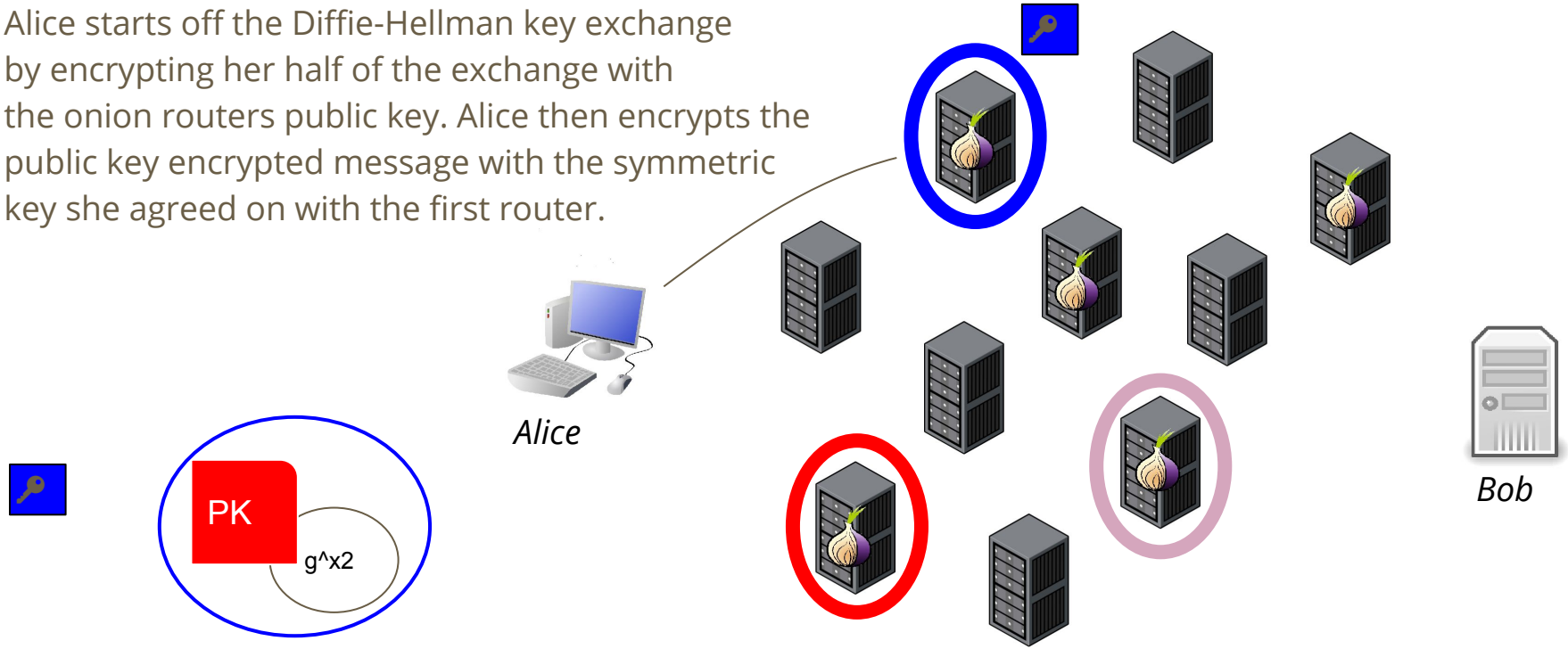
# Creating A Secure Circuit in the Tor Network cont.

Alice now has a key she shares with the first onion router. She must now instruct the first node (entry node) to *extend the circuit.* This means the onion router must now perform the same steps acting as an intermediary node for Alice and the middle node.



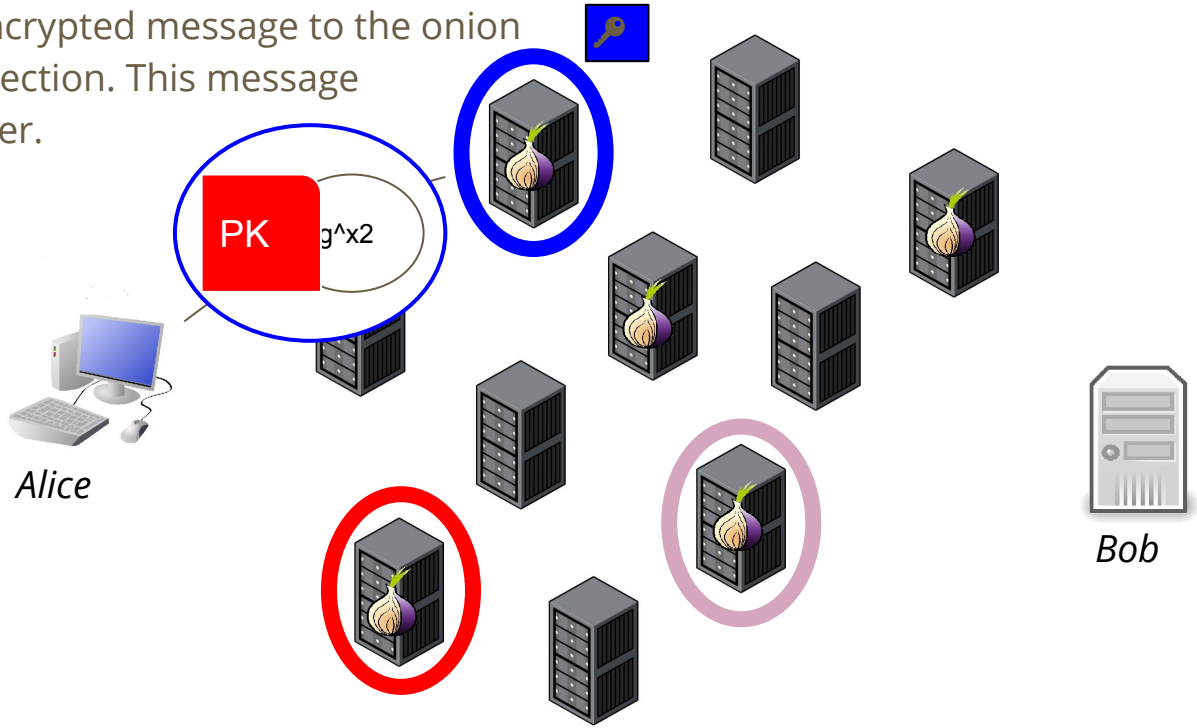
# Creating A Secure Circuit in the Tor Network cont.

Alice starts off the Diffie-Hellman key exchange by encrypting her half of the exchange with the onion routers public key. Alice then encrypts the public key encrypted message with the symmetric key she agreed on with the first router.



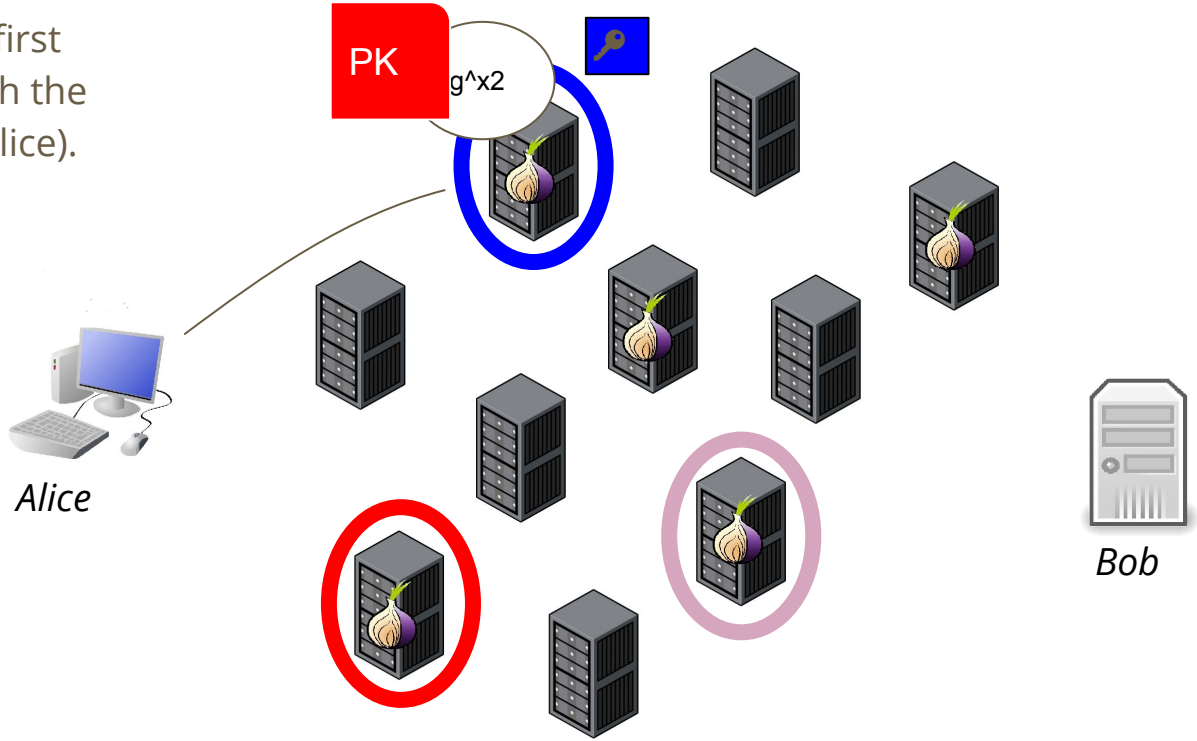
# Creating A Secure Circuit in the Tor Network cont.

Alice then sends the “double” encrypted message to the onion router over the secure TLS connection. This message is bound for the red circled router.



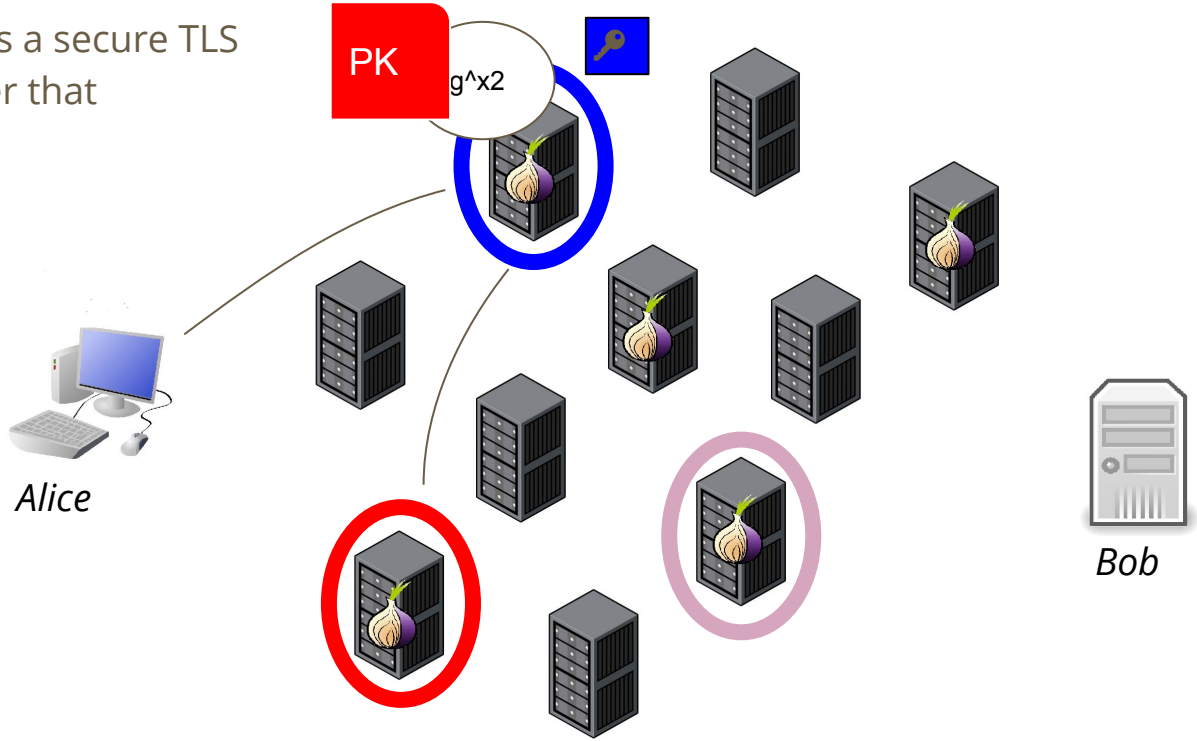
# Creating A Secure Circuit in the Tor Network cont.

The message is received by the first onion router and decrypted (with the symmetric key agreed on with Alice).



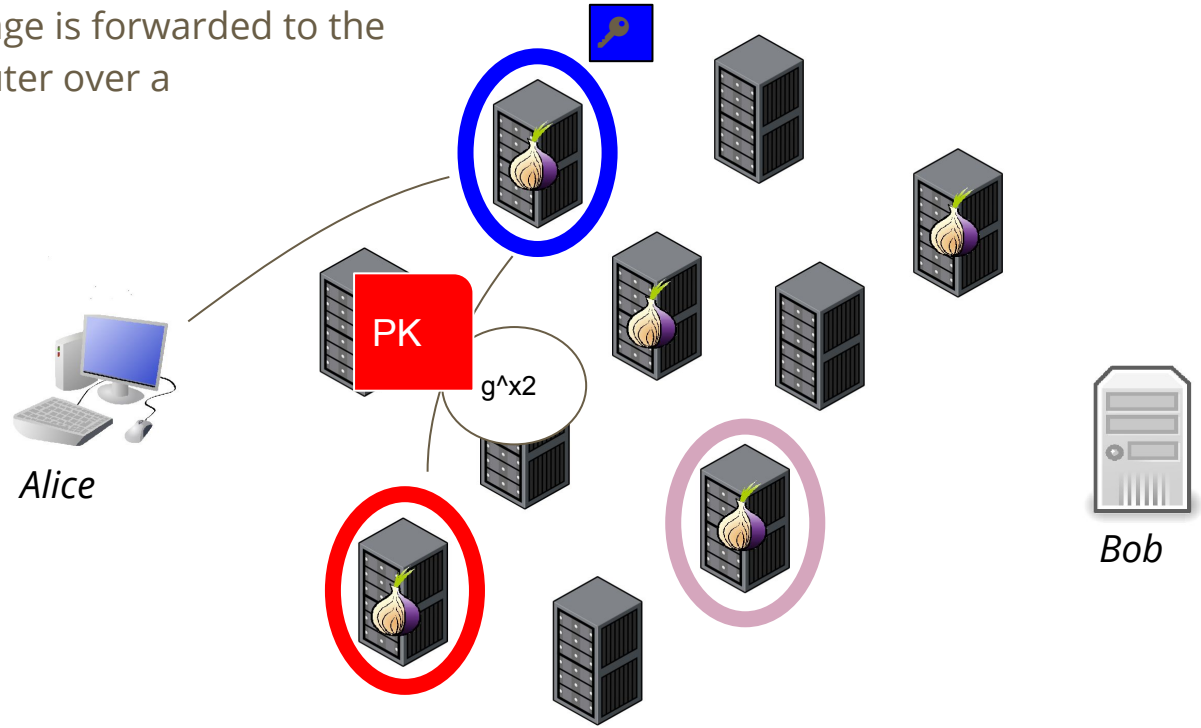
# Creating A Secure Circuit in the Tor Network cont.

The first onion router now forms a secure TLS connection with the onion router that Alice told it to.



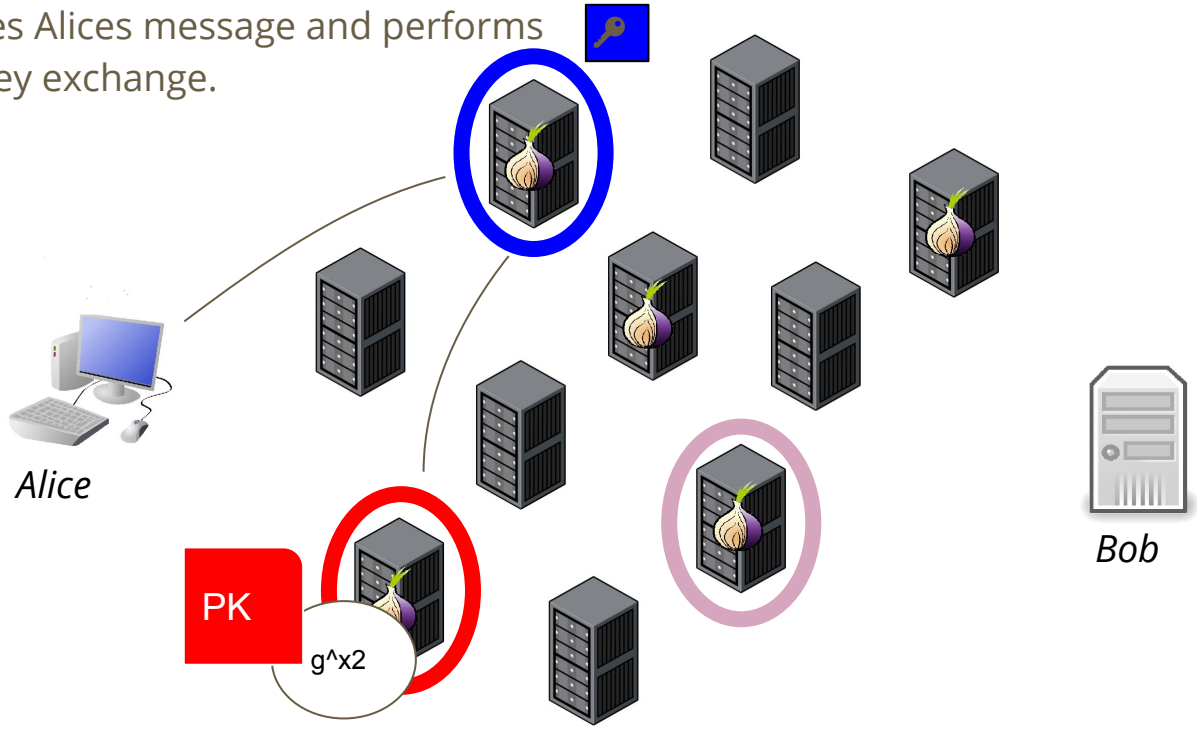
# Creating A Secure Circuit in the Tor Network cont.

The public key encrypted message is forwarded to the middle node - second onion router over a secure TLS connection.



# Creating A Secure Circuit in the Tor Network cont.

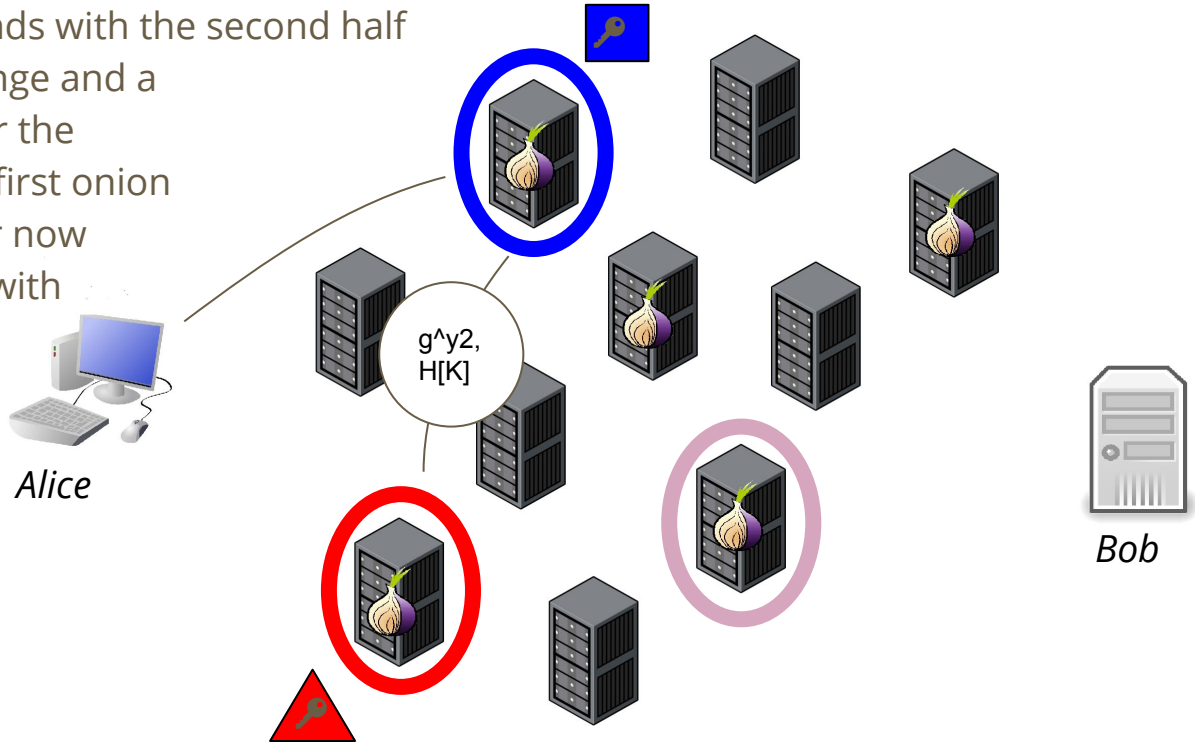
The second onion router receives Alices message and performs its share of the Diffie-Hellman key exchange.



# Creating A Secure Circuit in the Tor Network cont.

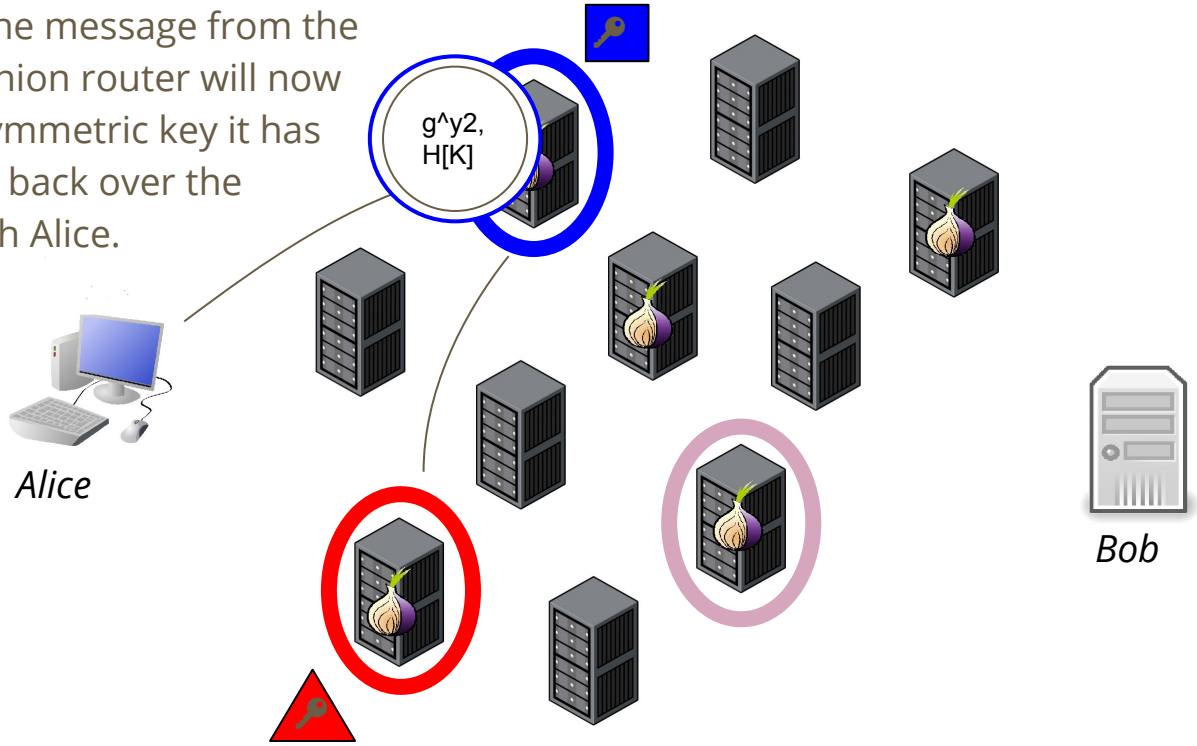
The second onion router responds with the second half of the Diffie-Hellman key exchange and a hash of the key. This is sent over the secure TLS connection with the first onion router. The second onion router now knows the key that it will share with Alice.

H[K] - Hash of key



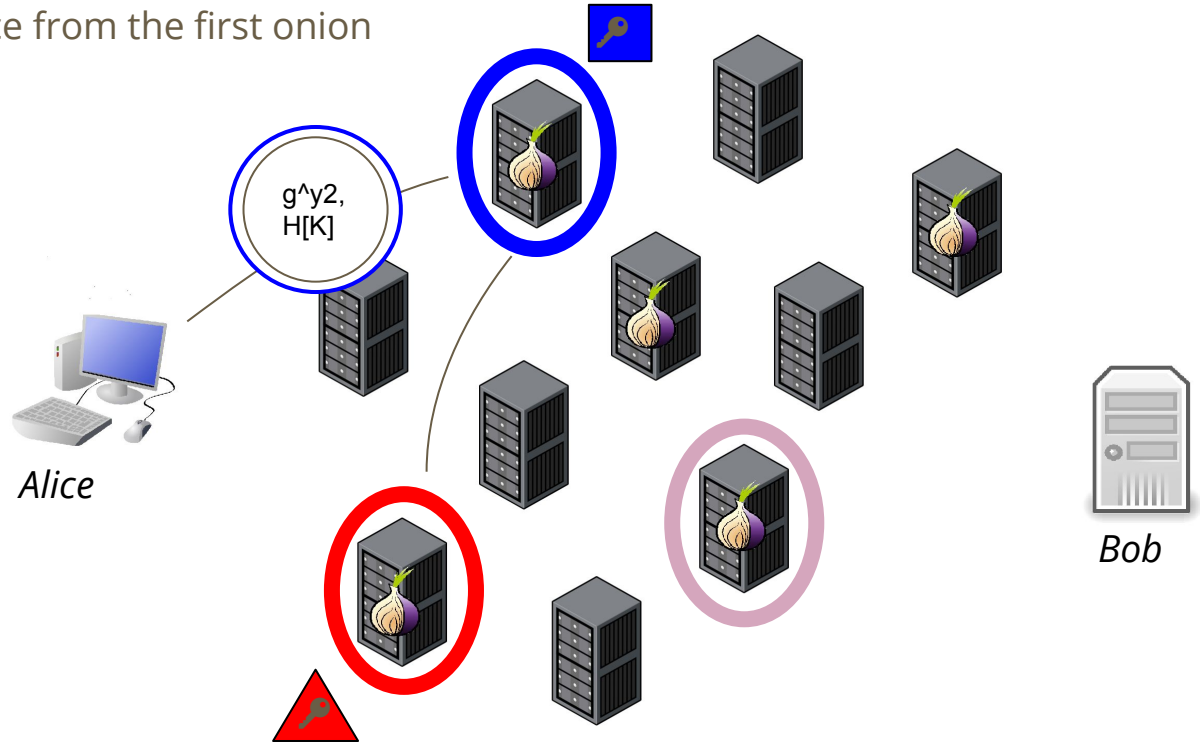
# Creating A Secure Circuit in the Tor Network cont.

The first onion router receives the message from the second onion router. The first onion router will now encrypt the message with the symmetric key it has agreed on with Alice and send it back over the secure TLS connection it has with Alice.



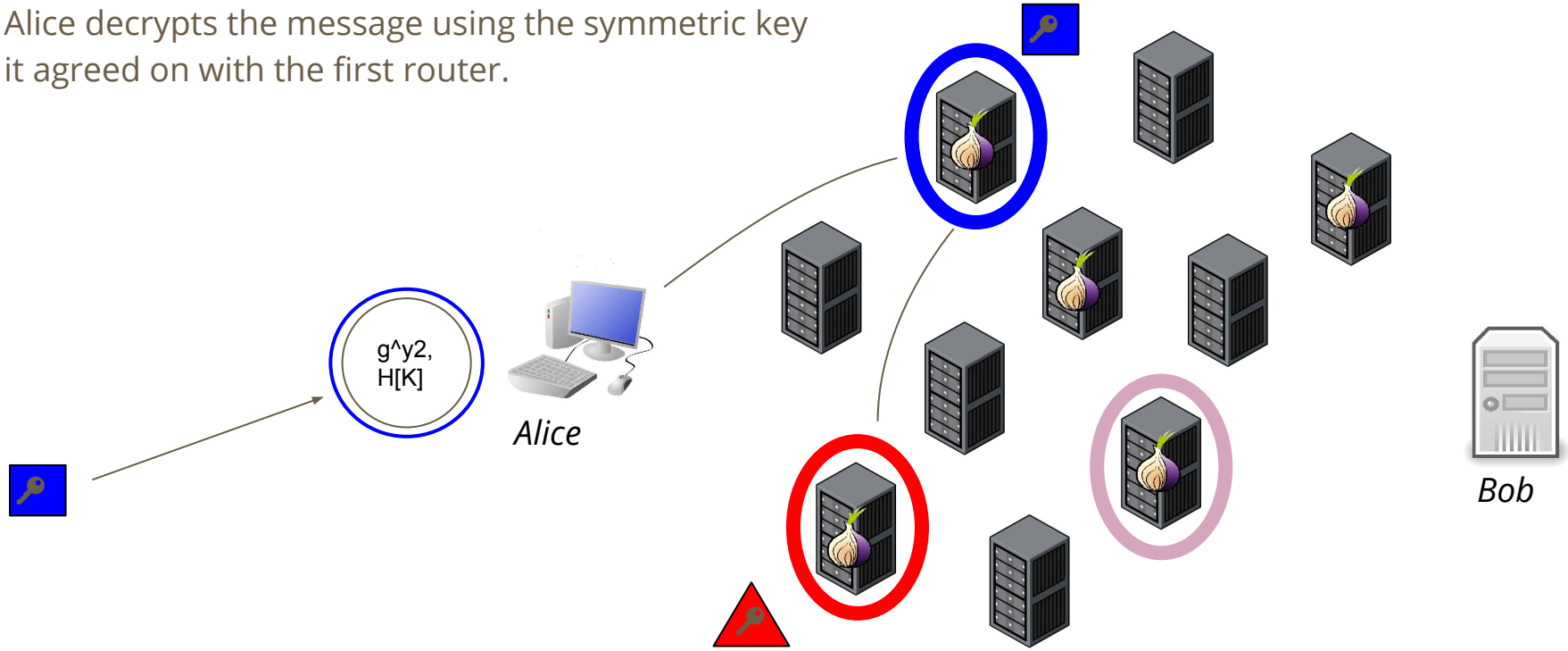
# Creating A Secure Circuit in the Tor Network cont.

The message is sent back to Alice from the first onion router.



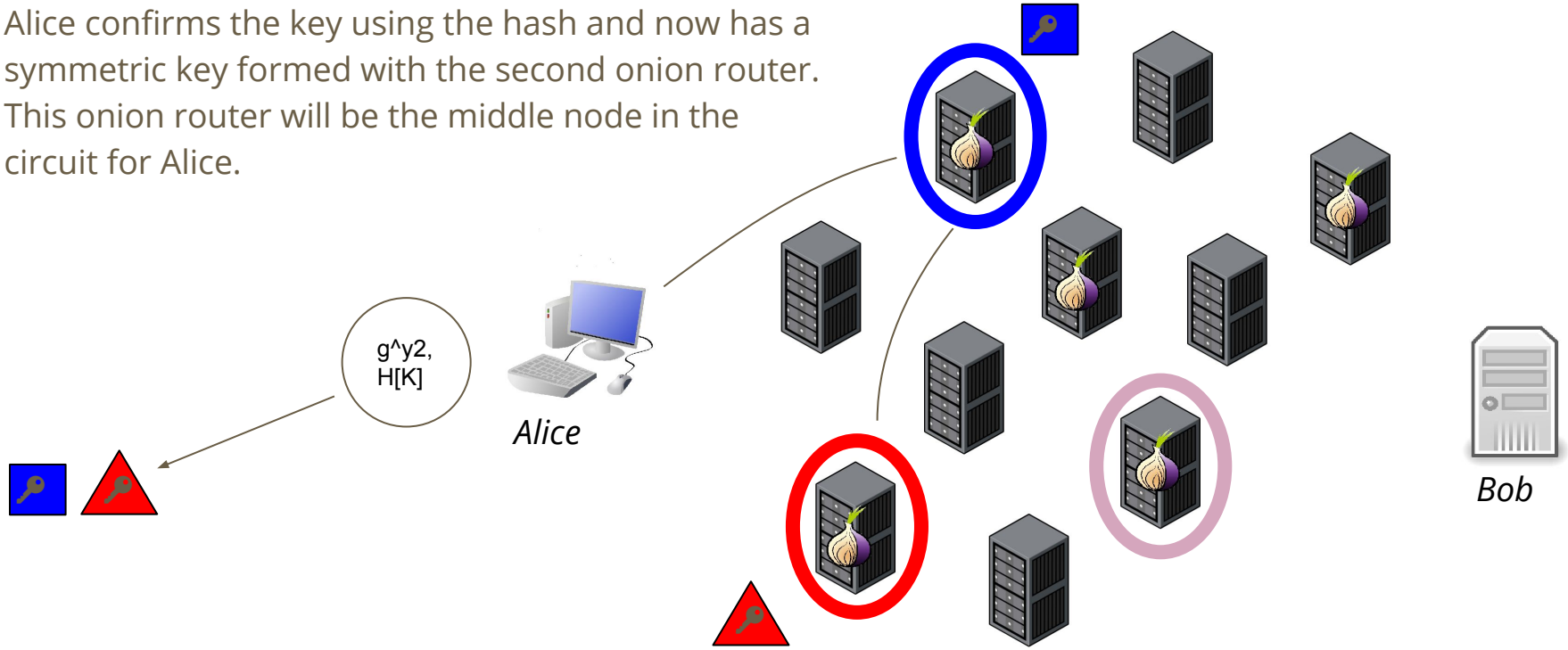
# Creating A Secure Circuit in the Tor Network cont.

Alice decrypts the message using the symmetric key it agreed on with the first router.



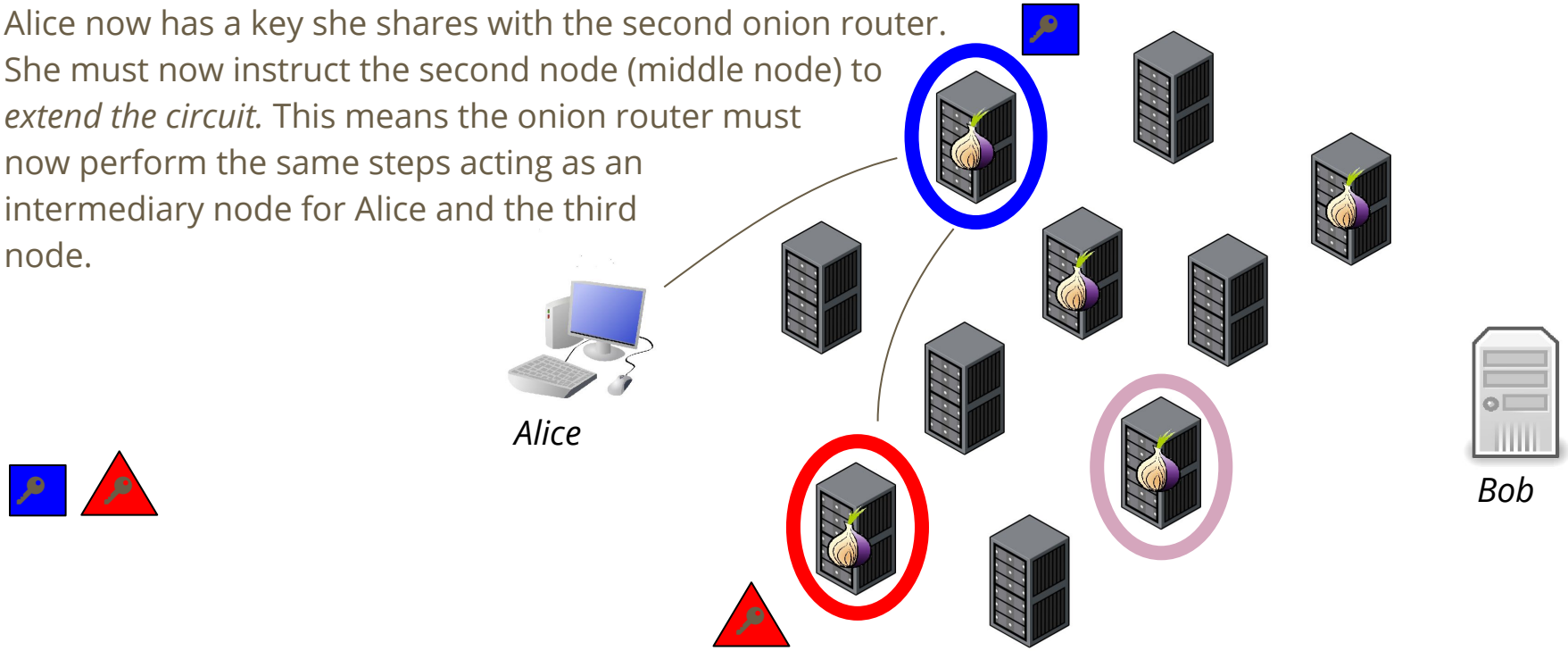
# Creating A Secure Circuit in the Tor Network cont.

Alice confirms the key using the hash and now has a symmetric key formed with the second onion router. This onion router will be the middle node in the circuit for Alice.



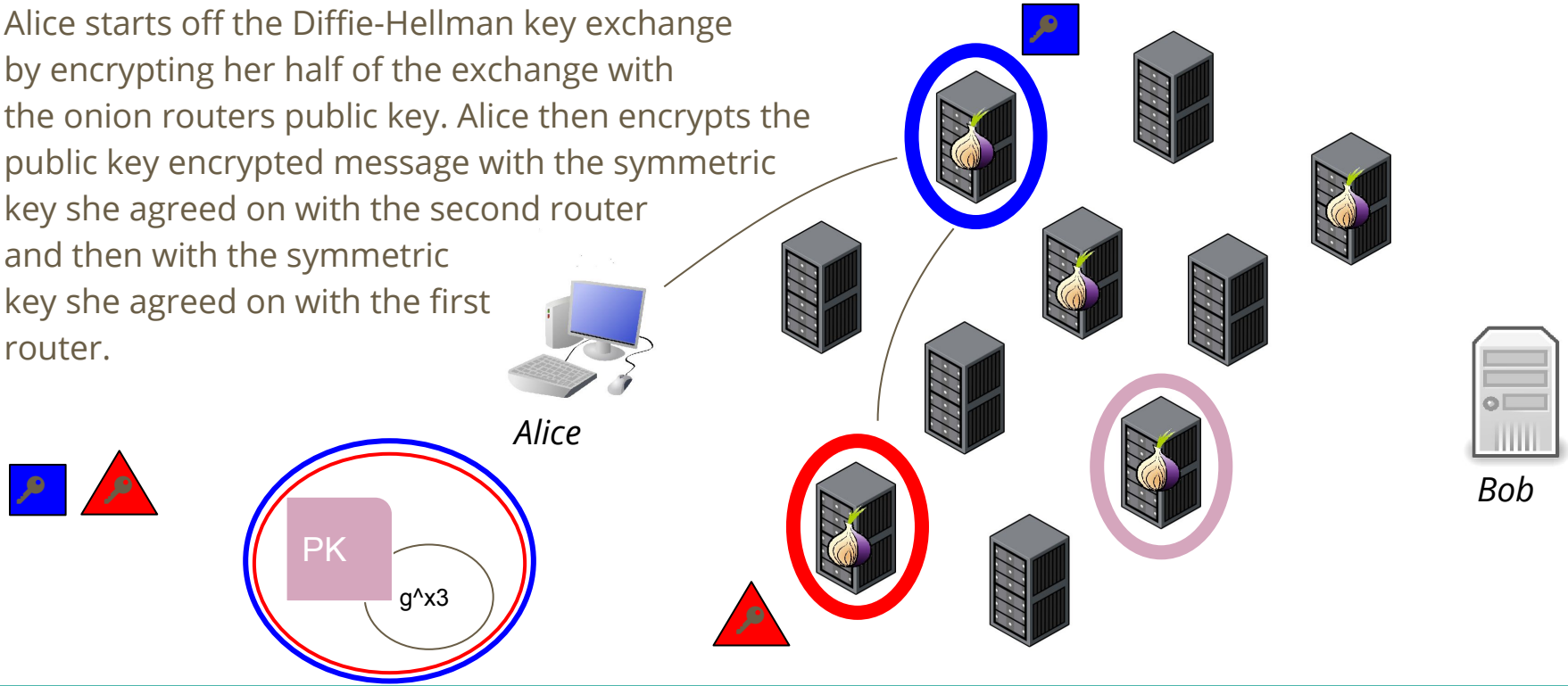
# Creating A Secure Circuit in the Tor Network cont.

Alice now has a key she shares with the second onion router. She must now instruct the second node (middle node) to *extend the circuit.* This means the onion router must now perform the same steps acting as an intermediary node for Alice and the third node.



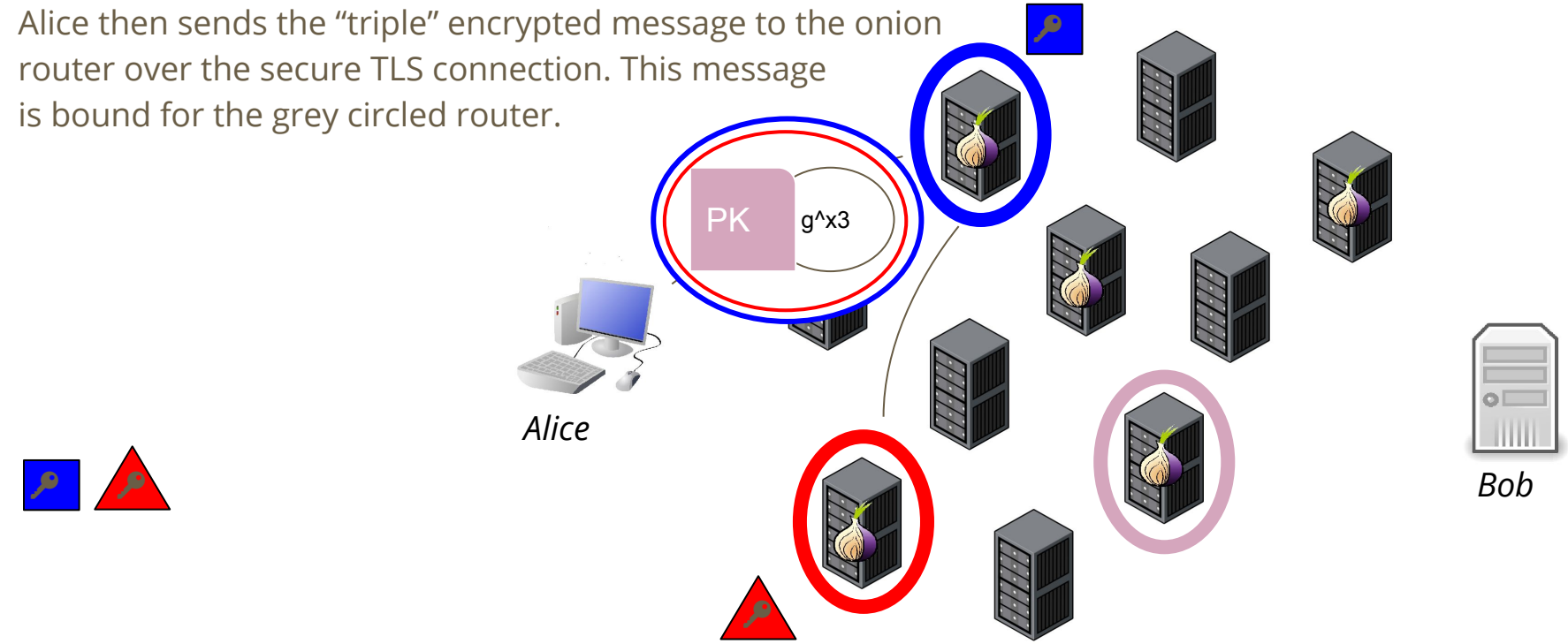
# Creating A Secure Circuit in the Tor Network cont.

Alice starts off the Diffie-Hellman key exchange by encrypting her half of the exchange with the onion routers public key. Alice then encrypts the public key encrypted message with the symmetric key she agreed on with the second router and then with the symmetric key she agreed on with the first router.



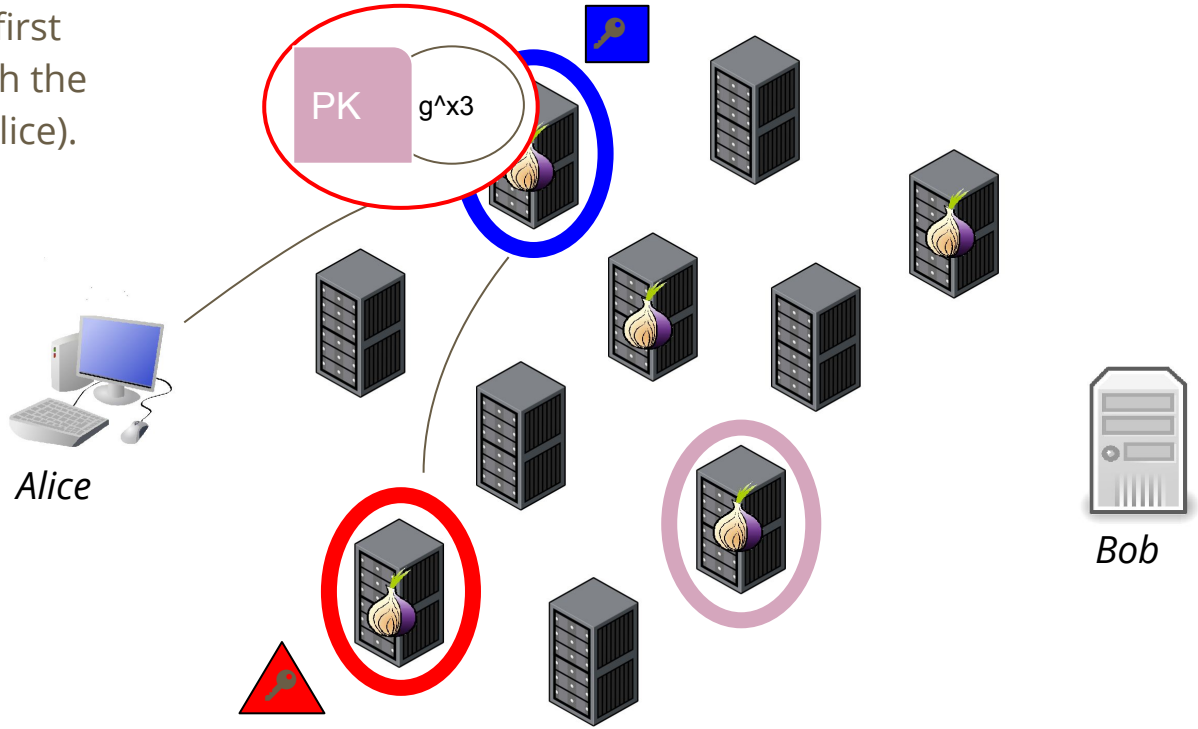
# Creating A Secure Circuit in the Tor Network cont.

Alice then sends the “triple” encrypted message to the onion router over the secure TLS connection. This message is bound for the grey circled router.



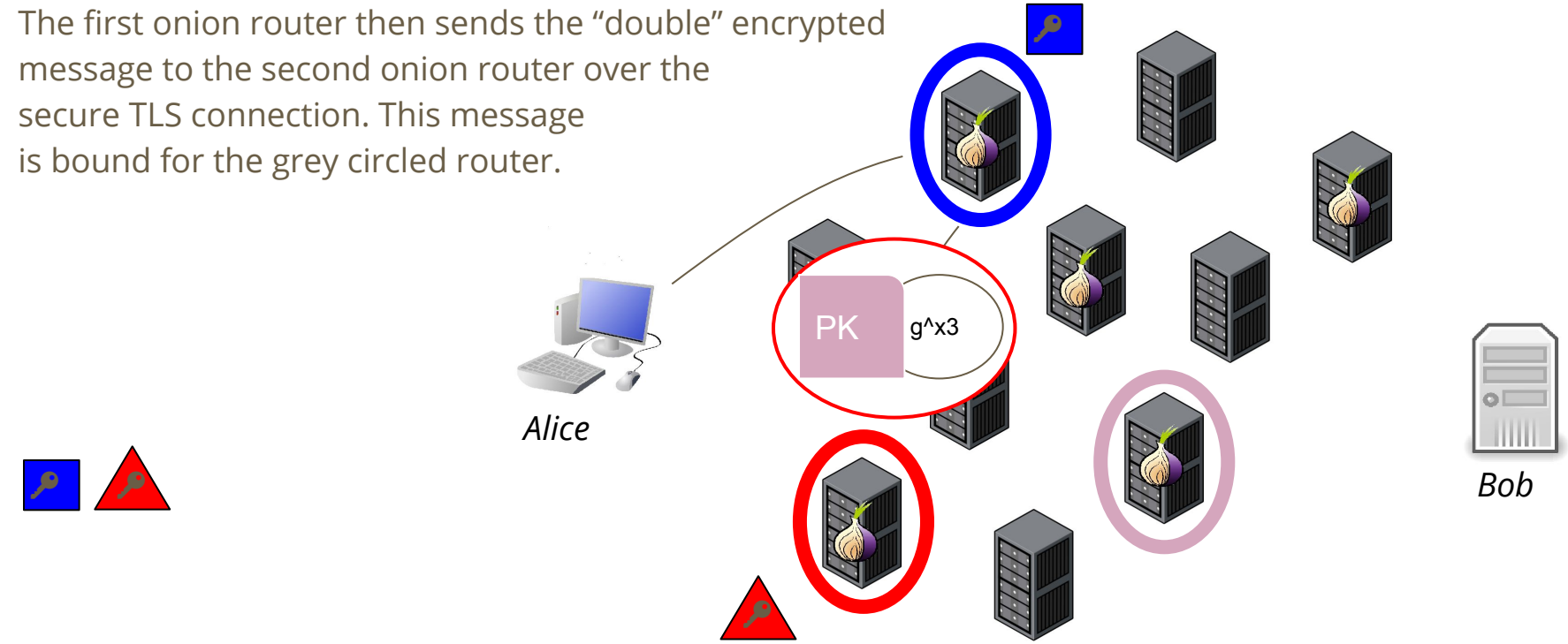
# Creating A Secure Circuit in the Tor Network cont.

The message is received by the first onion router and decrypted (with the symmetric key agreed on with Alice).



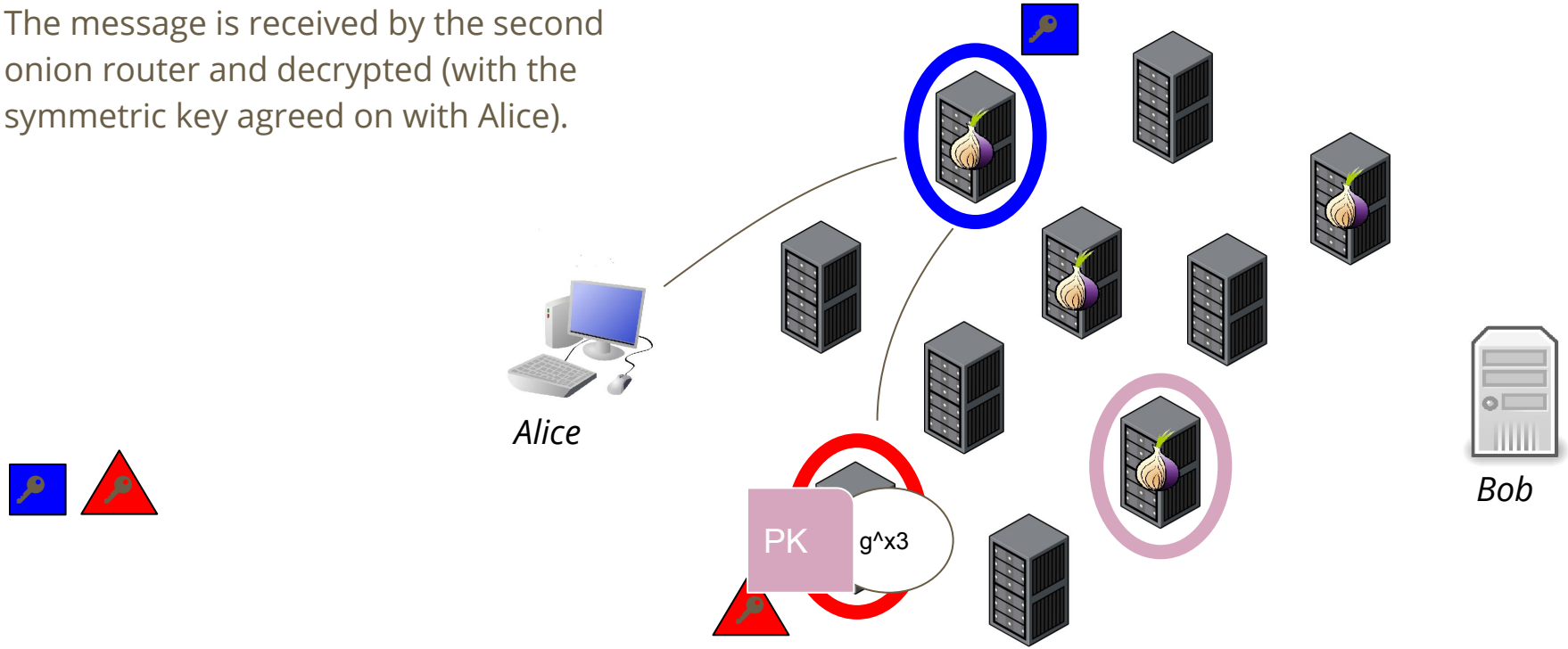
# Creating A Secure Circuit in the Tor Network cont.

The first onion router then sends the “double” encrypted message to the second onion router over the secure TLS connection. This message is bound for the grey circled router.



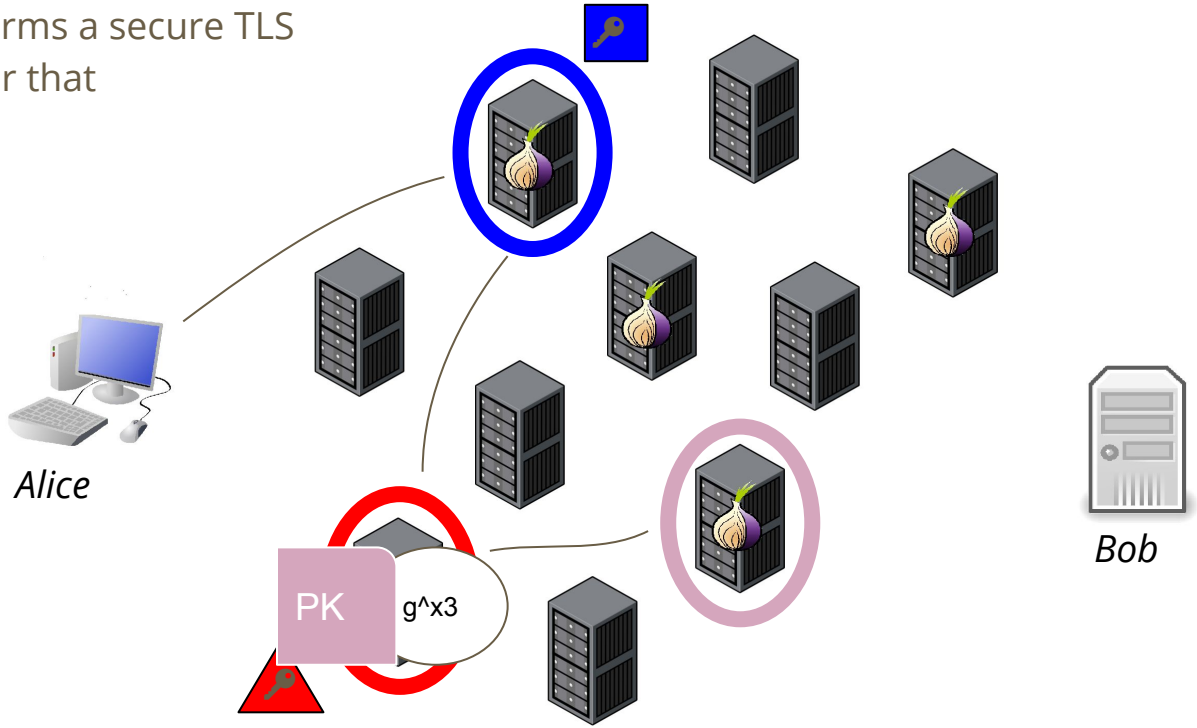
# Creating A Secure Circuit in the Tor Network cont.

The message is received by the second onion router and decrypted (with the symmetric key agreed on with Alice).



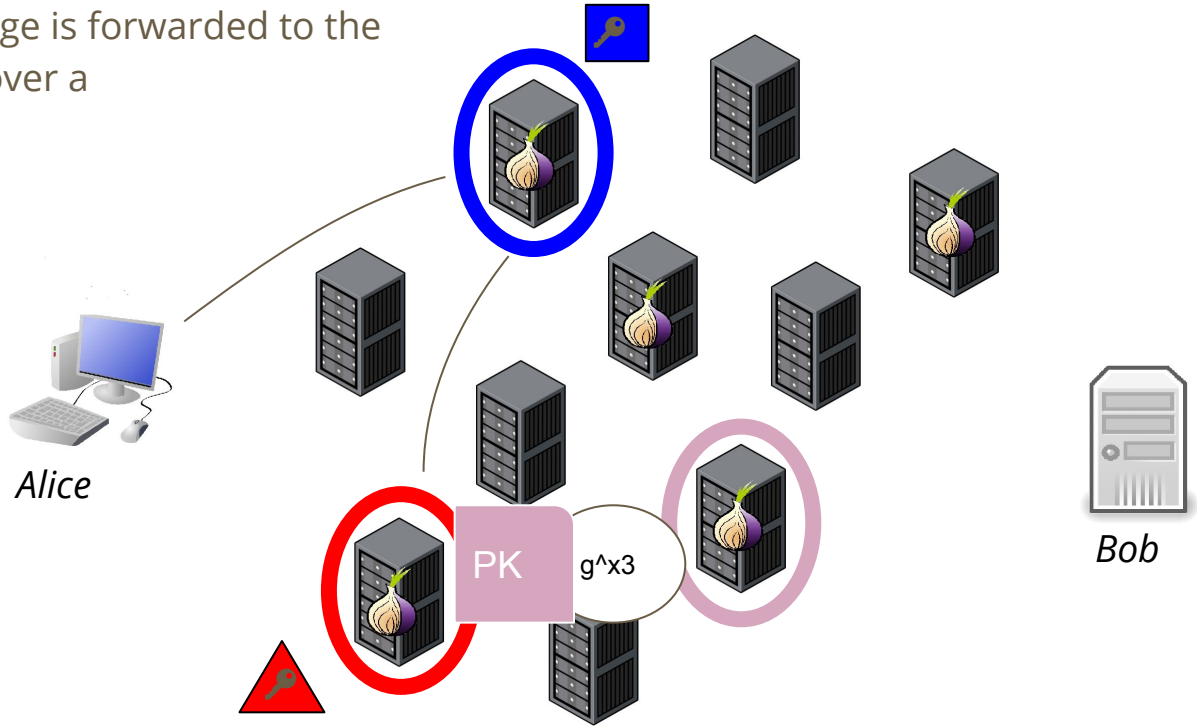
# Creating A Secure Circuit in the Tor Network cont.

The second onion router now forms a secure TLS connection with the onion router that Alice told it to.



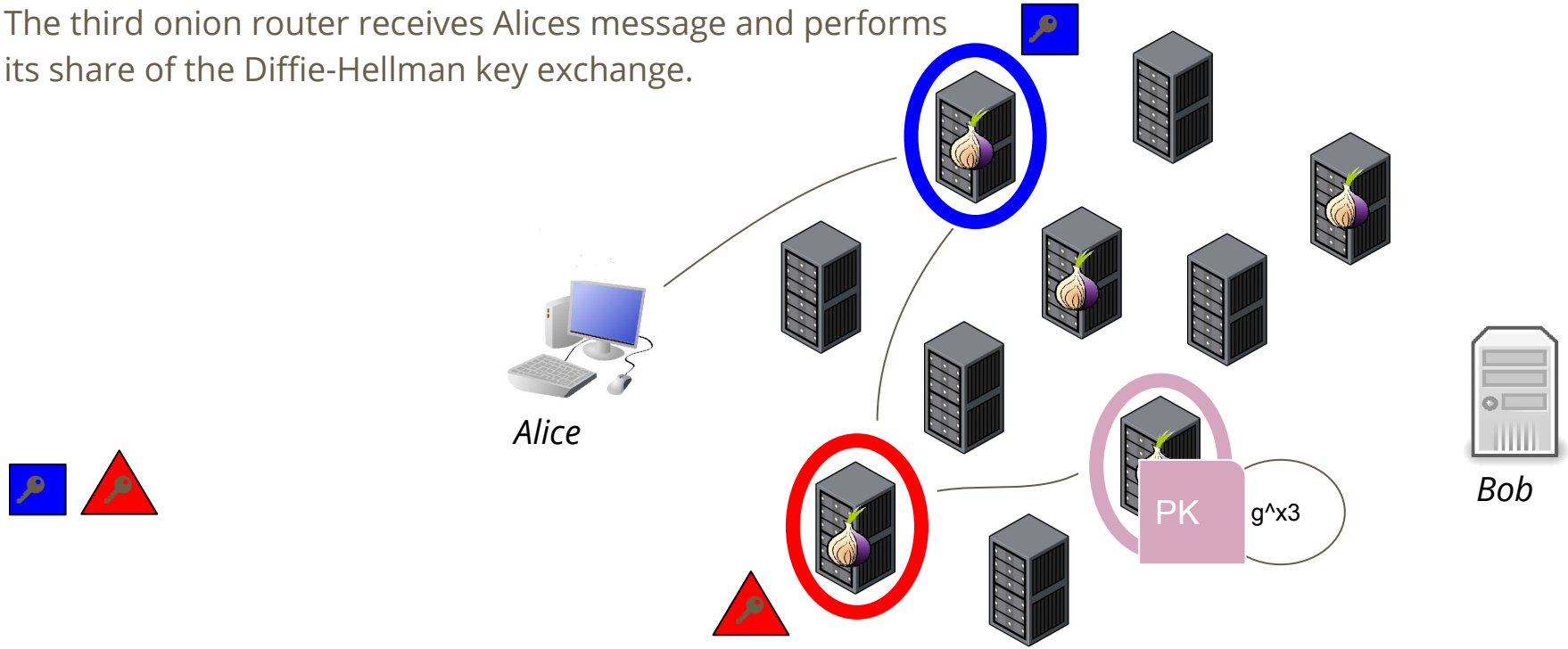
# Creating A Secure Circuit in the Tor Network cont.

The public key encrypted message is forwarded to the third node - third onion router over a secure TLS connection.



# Creating A Secure Circuit in the Tor Network cont.

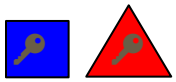
The third onion router receives Alices message and performs its share of the Diffie-Hellman key exchange.

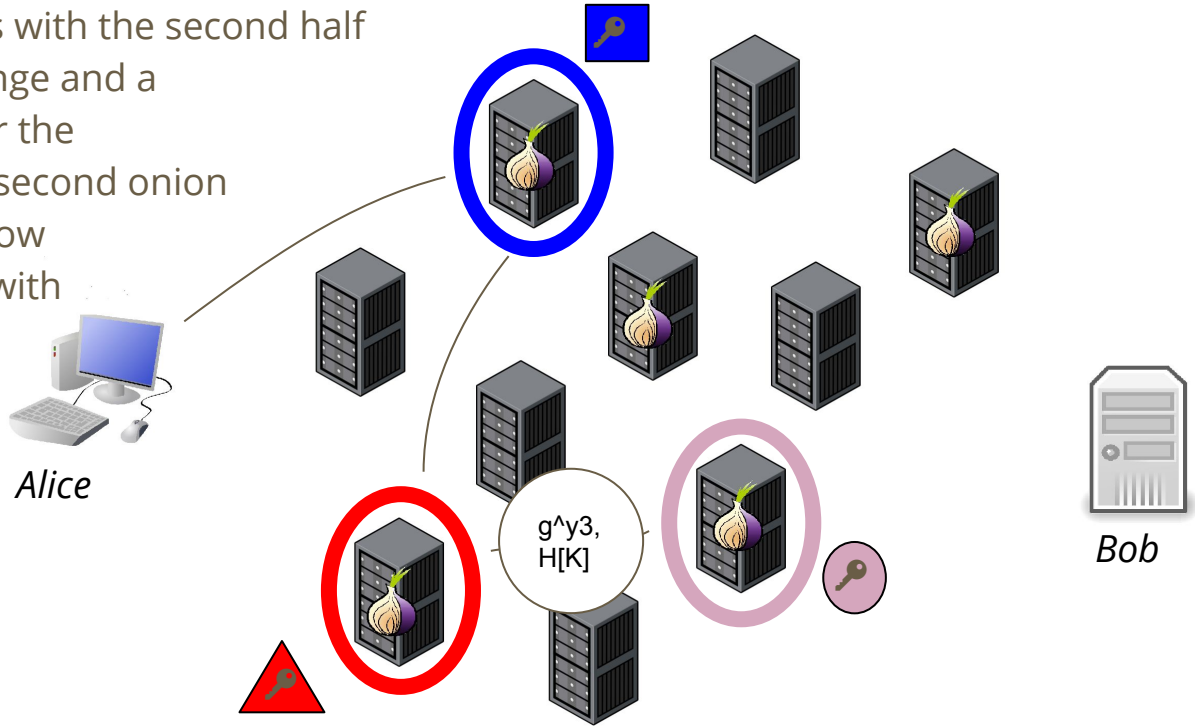


# Creating A Secure Circuit in the Tor Network cont.

The third onion router responds with the second half of the Diffie-Hellman key exchange and a hash of the key. This is sent over the secure TLS connection with the second onion router. The third onion router now knows the key that it will share with Alice.

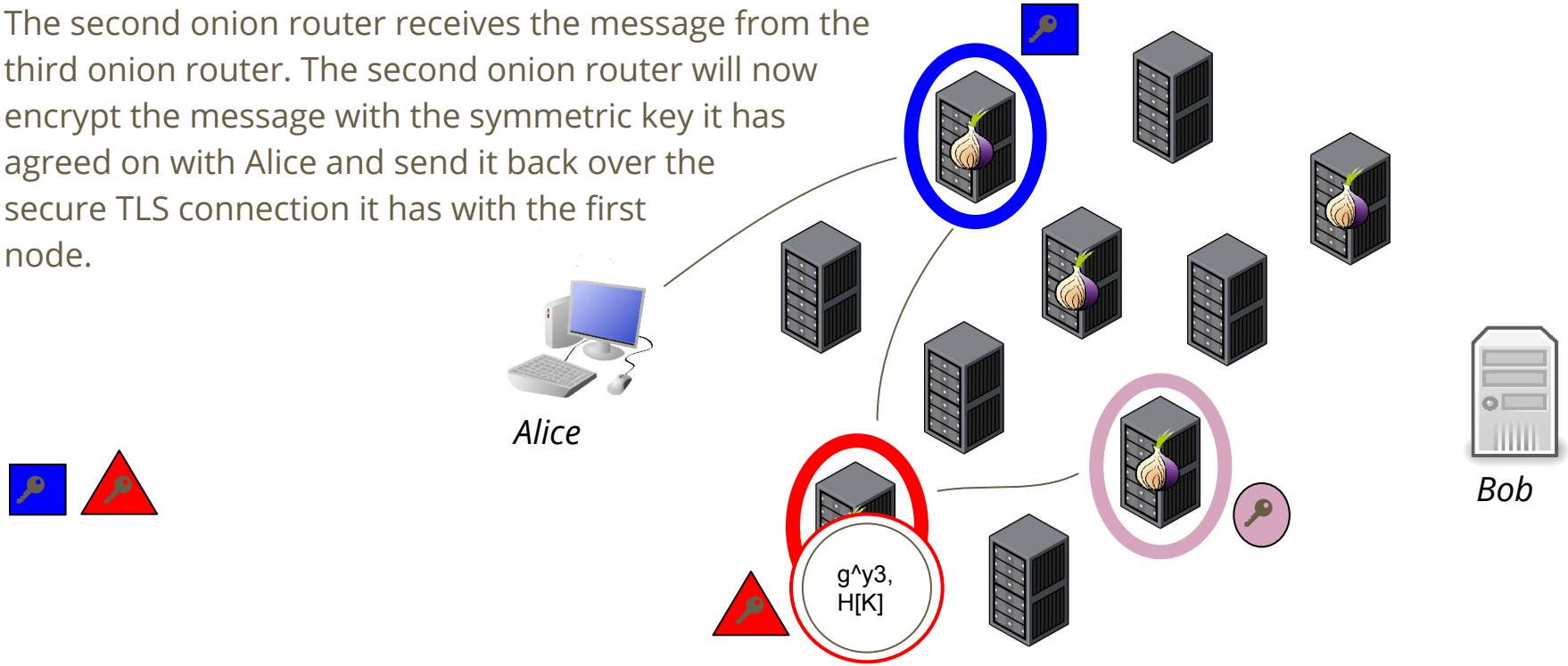
H[K] - Hash of key





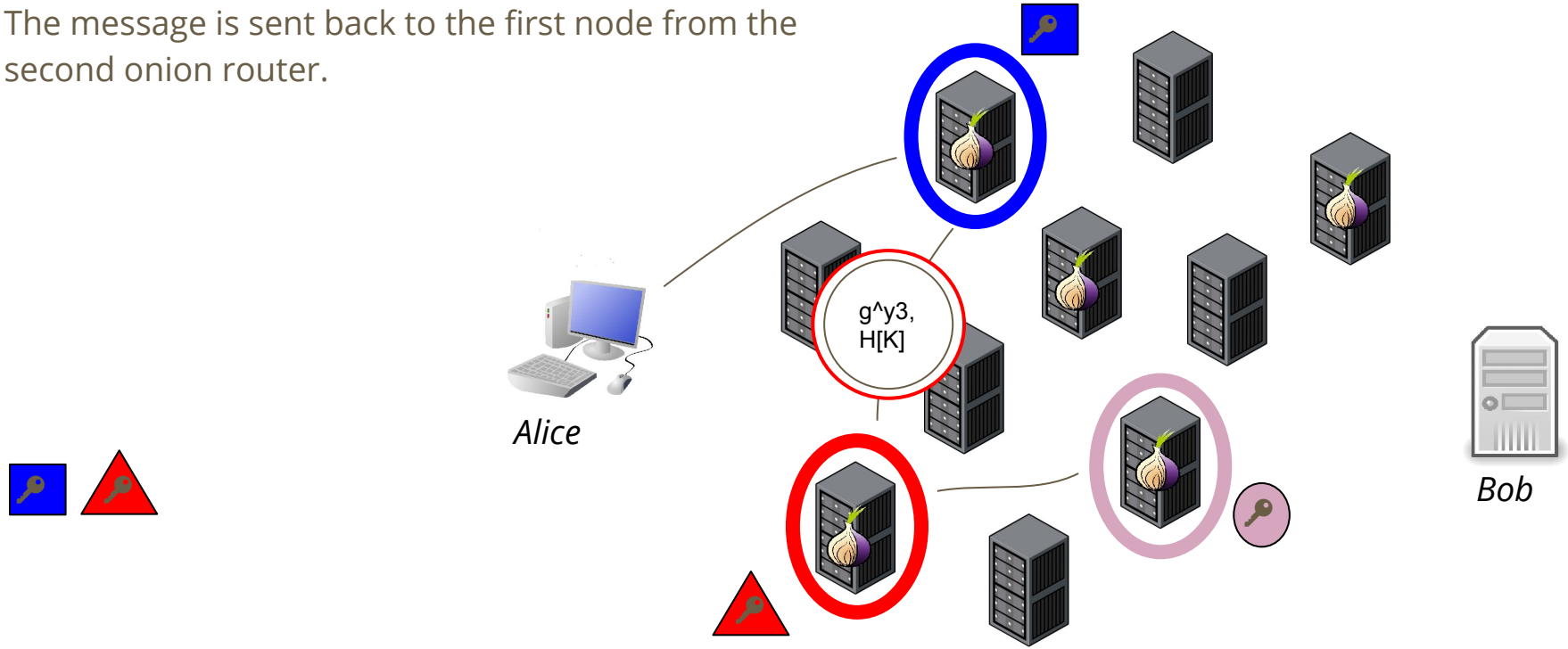
# Creating A Secure Circuit in the Tor Network cont.

The second onion router receives the message from the third onion router. The second onion router will now encrypt the message with the symmetric key it has agreed on with Alice and send it back over the secure TLS connection it has with the first node.



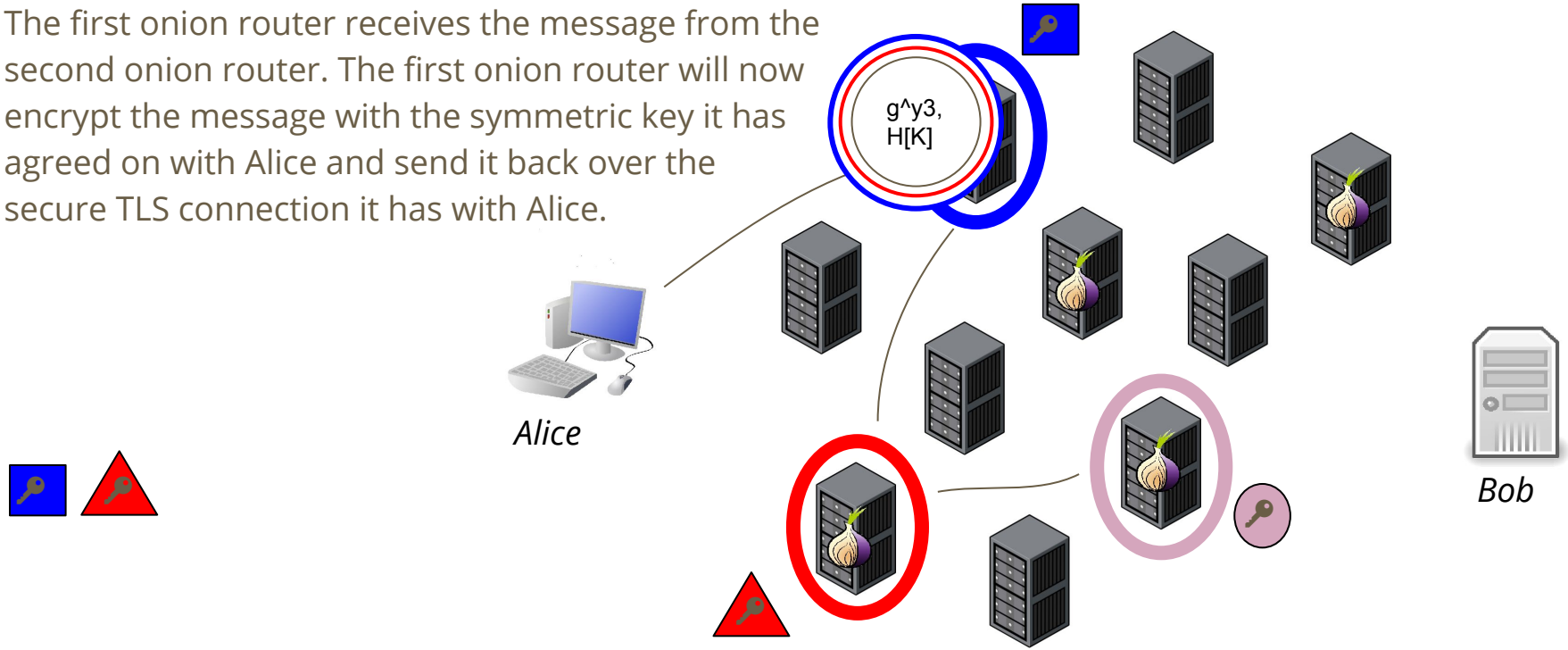
# Creating A Secure Circuit in the Tor Network cont.

The message is sent back to the first node from the second onion router.



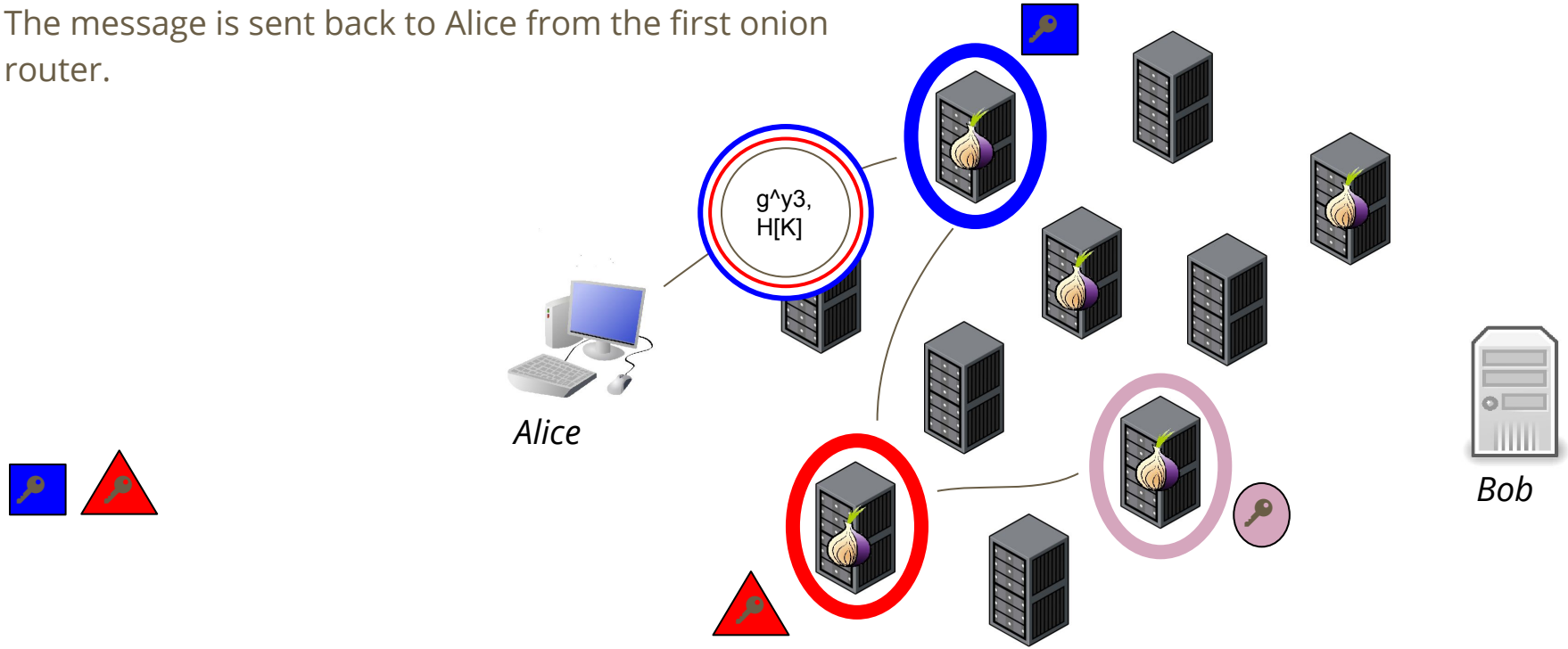
# Creating A Secure Circuit in the Tor Network cont.

The first onion router receives the message from the second onion router. The first onion router will now encrypt the message with the symmetric key it has agreed on with Alice and send it back over the secure TLS connection it has with Alice.



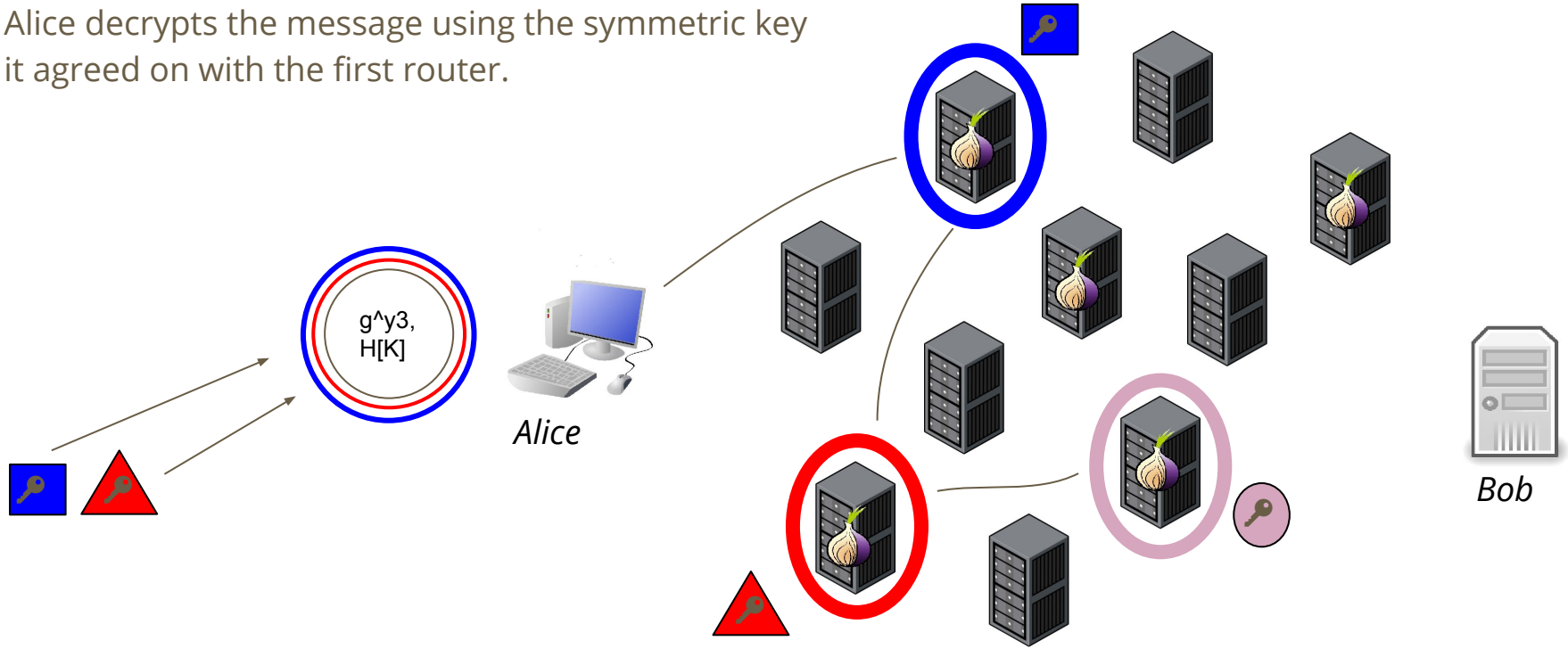
# Creating A Secure Circuit in the Tor Network cont.

The message is sent back to Alice from the first onion router.



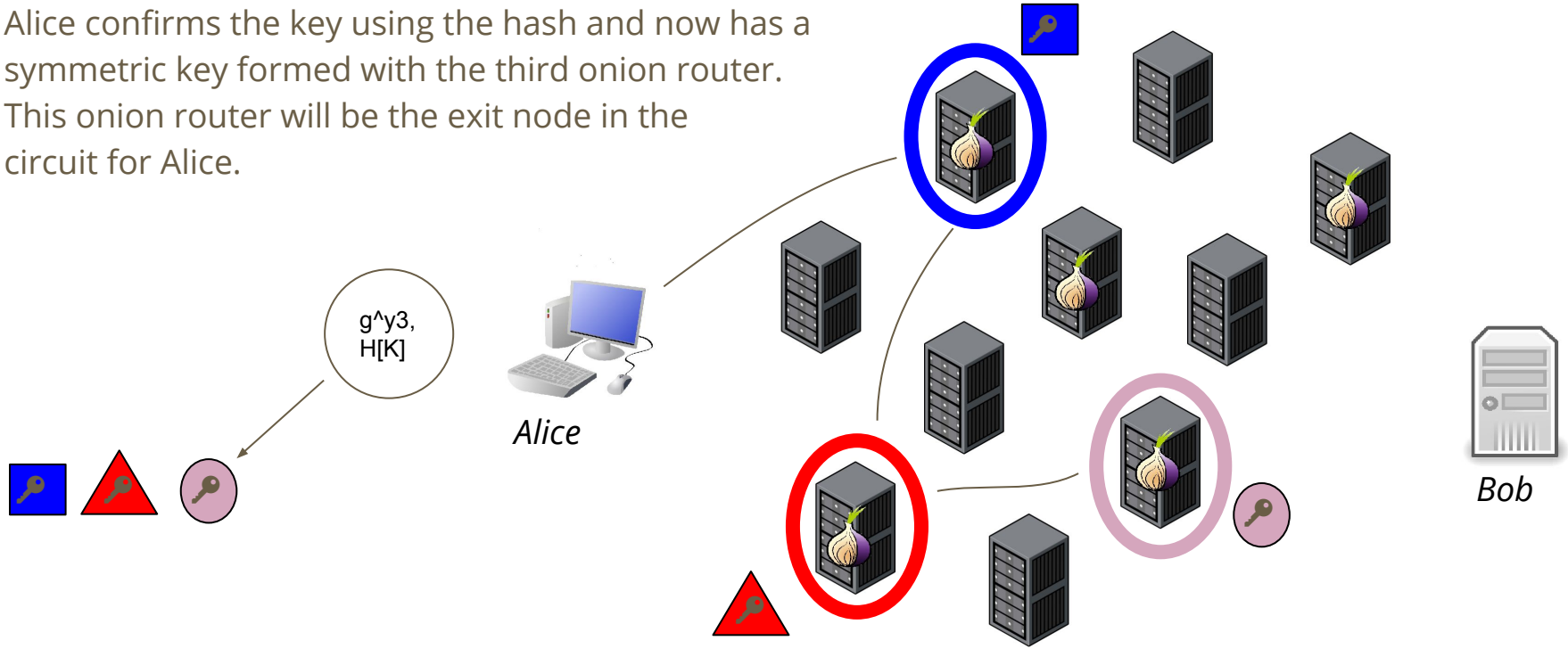
# Creating A Secure Circuit in the Tor Network cont.

Alice decrypts the message using the symmetric key it agreed on with the first router.



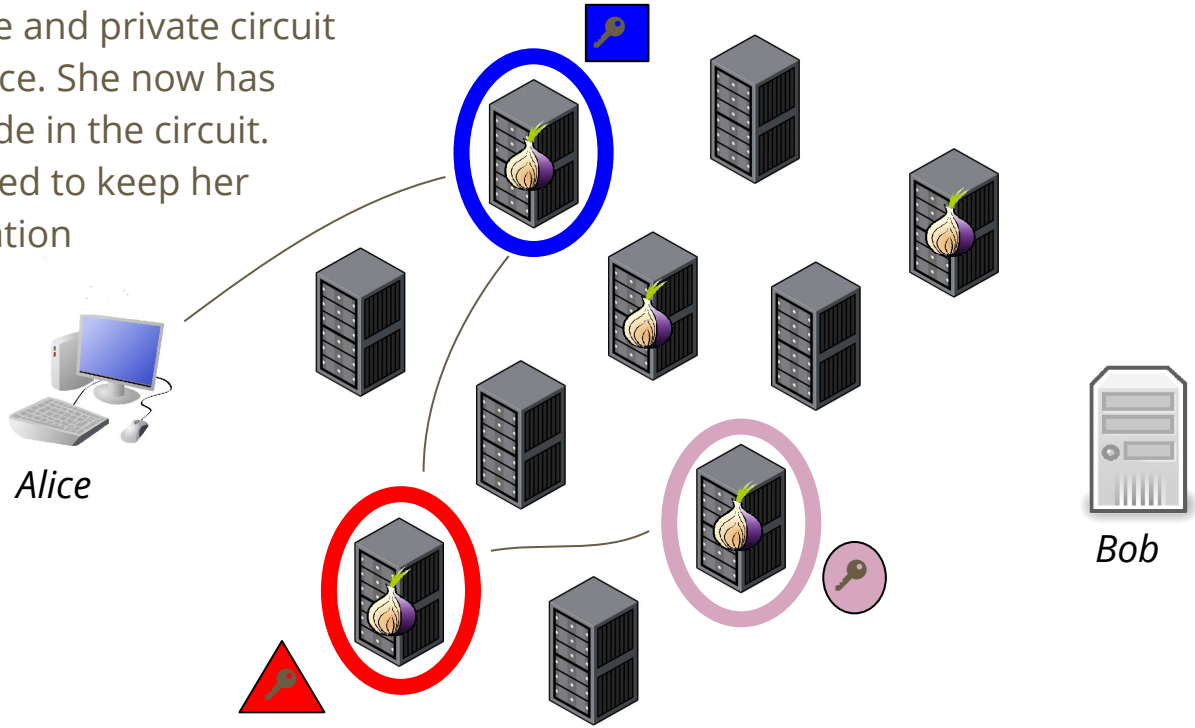
# Creating A Secure Circuit in the Tor Network cont.

Alice confirms the key using the hash and now has a symmetric key formed with the third onion router. This onion router will be the exit node in the circuit for Alice.



# Creating A Secure Circuit in the Tor Network fin.

Alice has now formed her secure and private circuit to (e.g.) connect to a onion service. She now has 3 symmetric keys, 1 for each node in the circuit. These symmetric keys will be used to keep her communication with the destination secure and private.



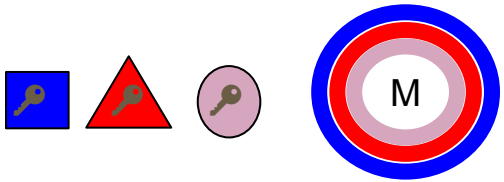
# How Tor works cont.

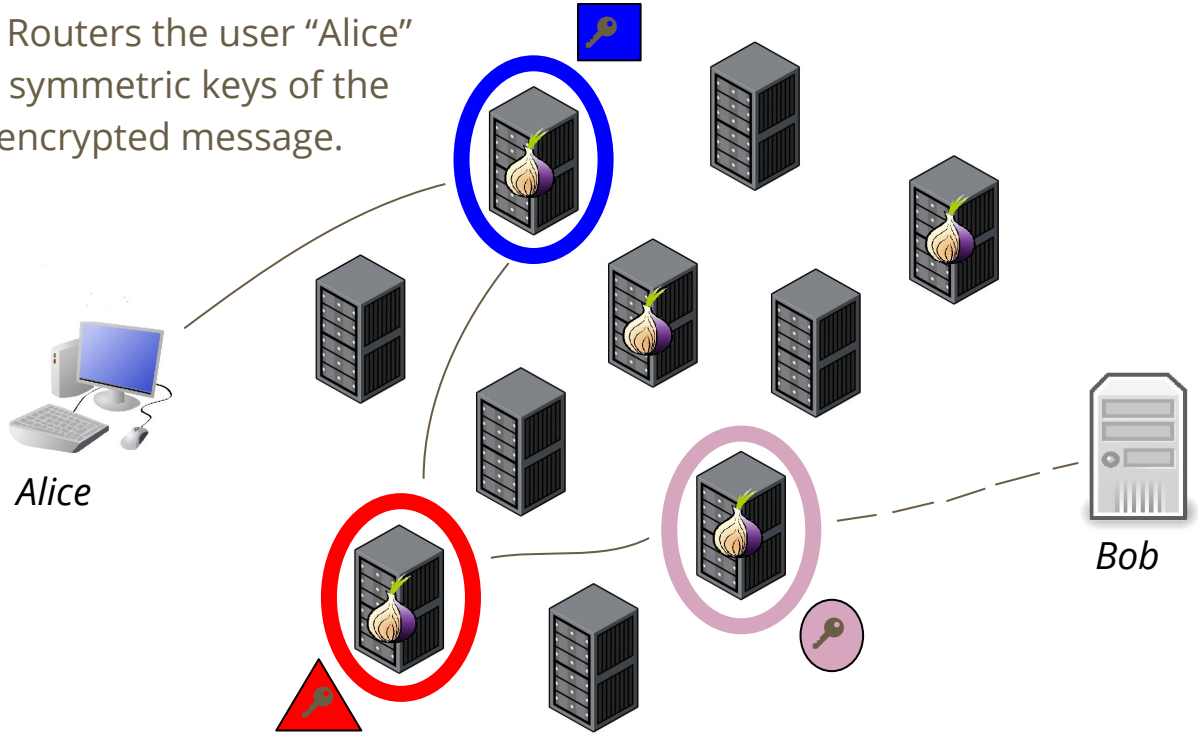
**Step 2)** With the selected Onion Routers the user “Alice” encrypts their message with the symmetric keys of the onion routers into a triple layer encrypted message.

Straight line - encrypted

Dashed line - unencrypted

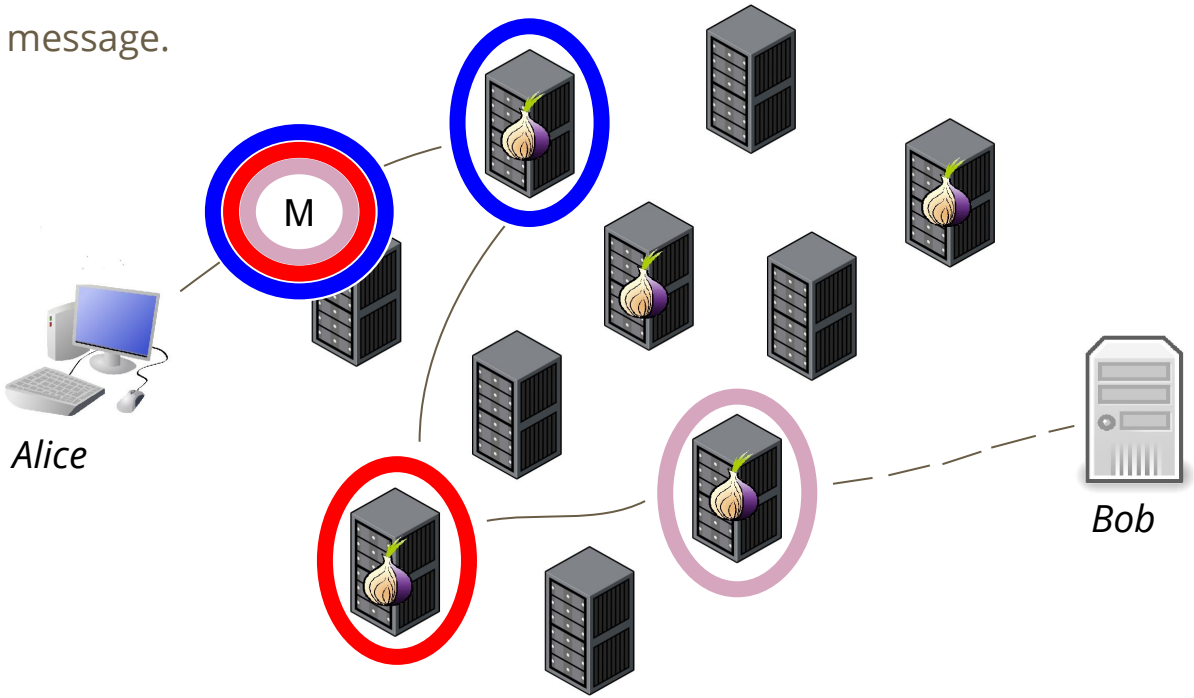
M - message (e.g. packet)



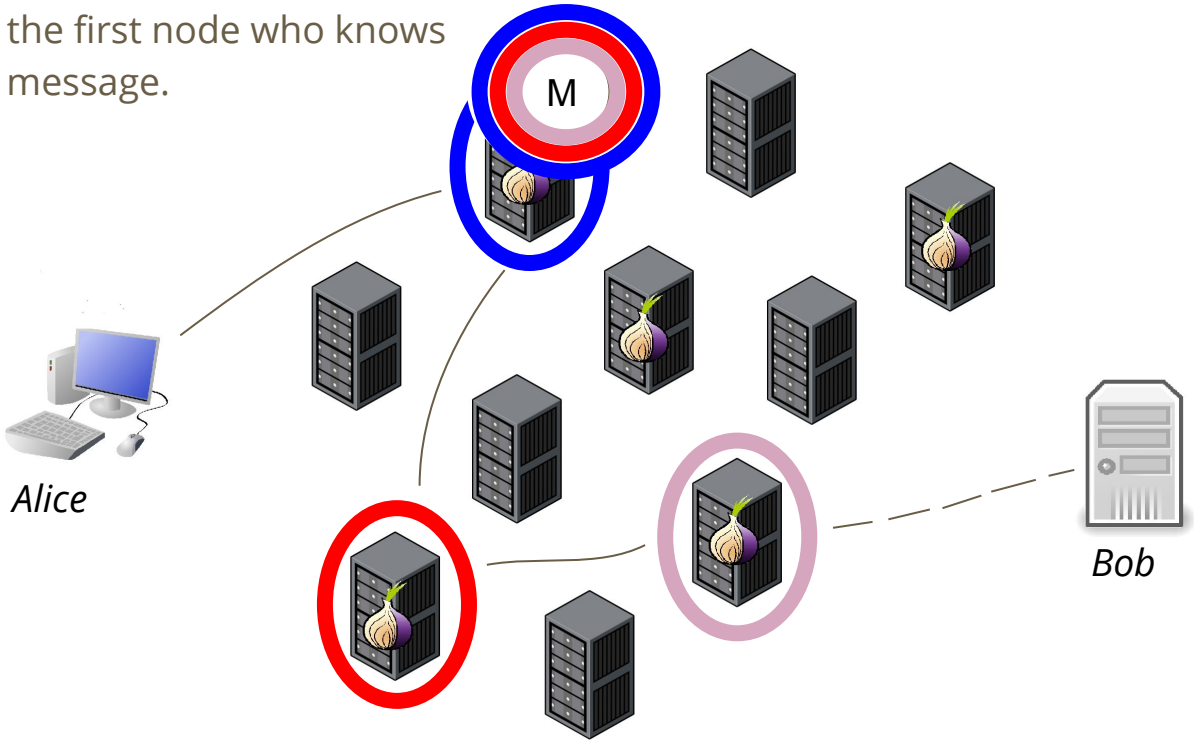
# How Tor works cont.

**Step 3a)** The message travels to the first node who knows Alice sent the 3-layer encryption message.



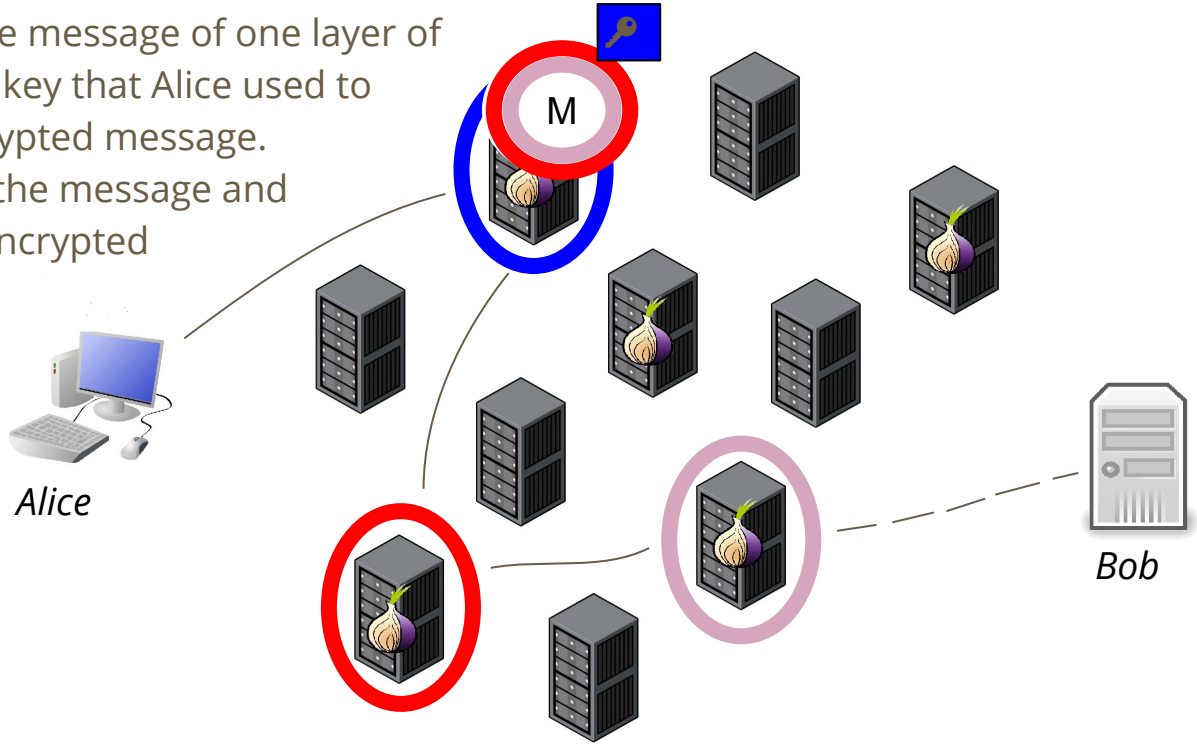
# How Tor works cont.

**Step 3a)** The message travels to the first node who knows Alice sent the 3-layer encryption message.



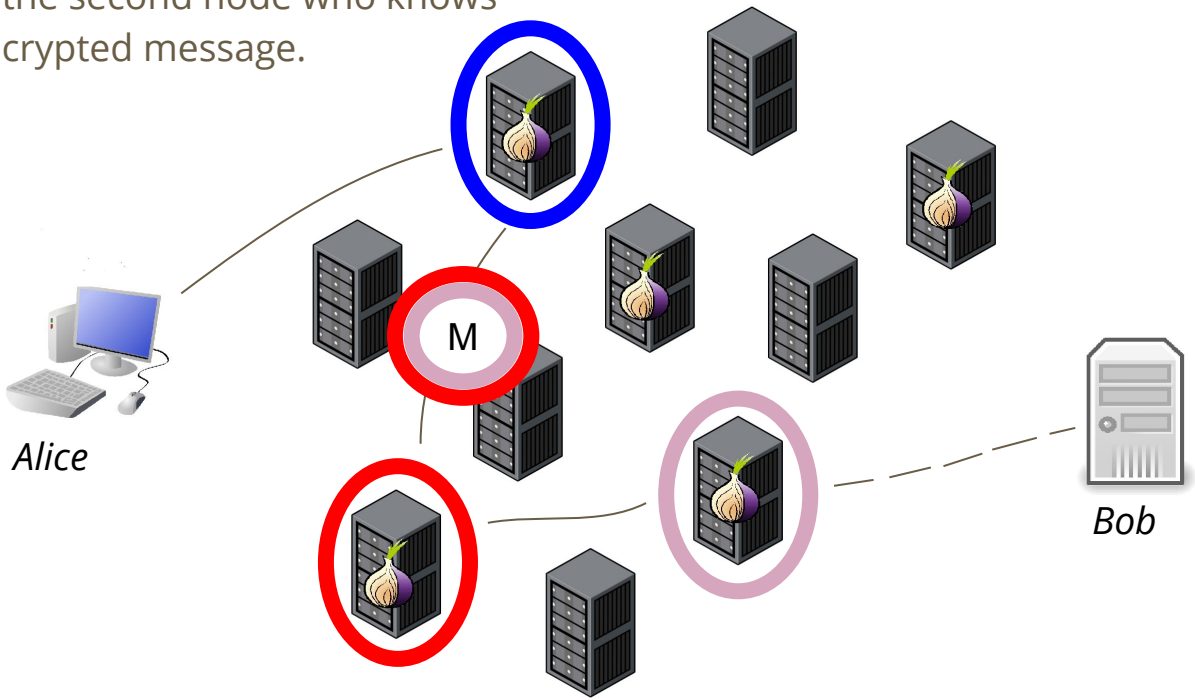
# How Tor works cont.

**Step 3b)** The first node strips the message of one layer of encryption using the symmetric key that Alice used to encrypt the already 2-layer encrypted message. The first node knows Alice sent the message and now forwards the two layered encrypted message to the second node.



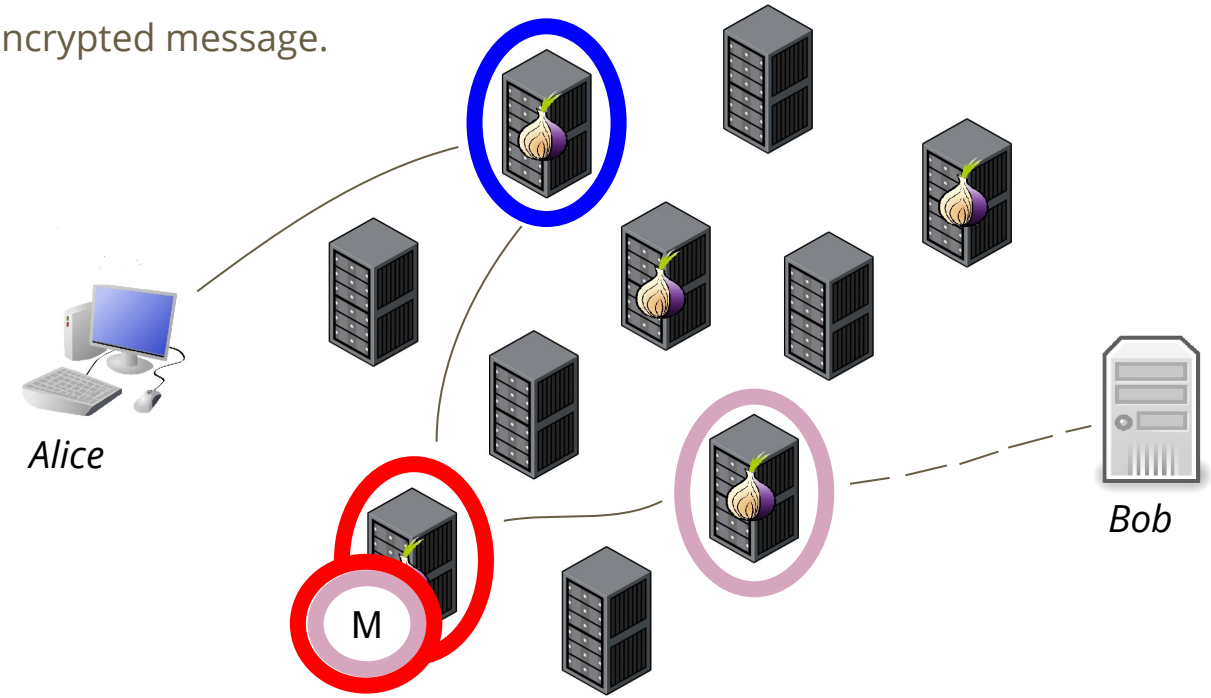
# How Tor works cont.

**Step 4a)** The message travels to the second node who knows the first node sent the 2-layer encrypted message.



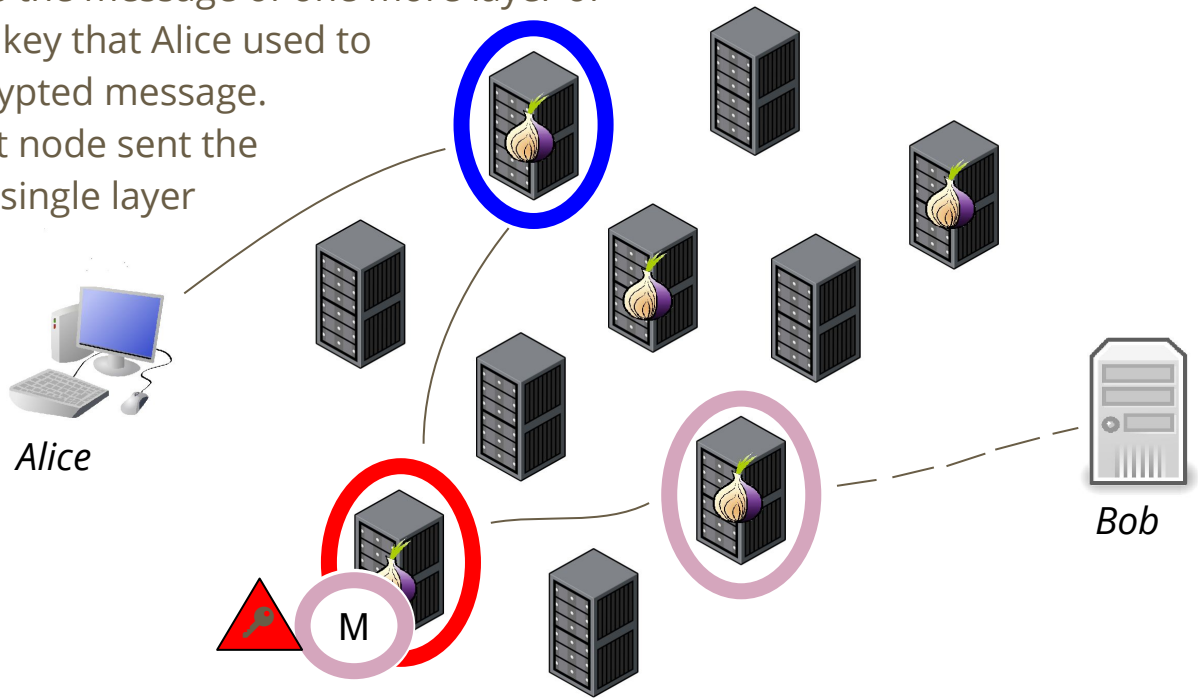
# How Tor works cont.

**Step 4a)** The message travels to the second node who knows the first node sent the 2-layer encrypted message.



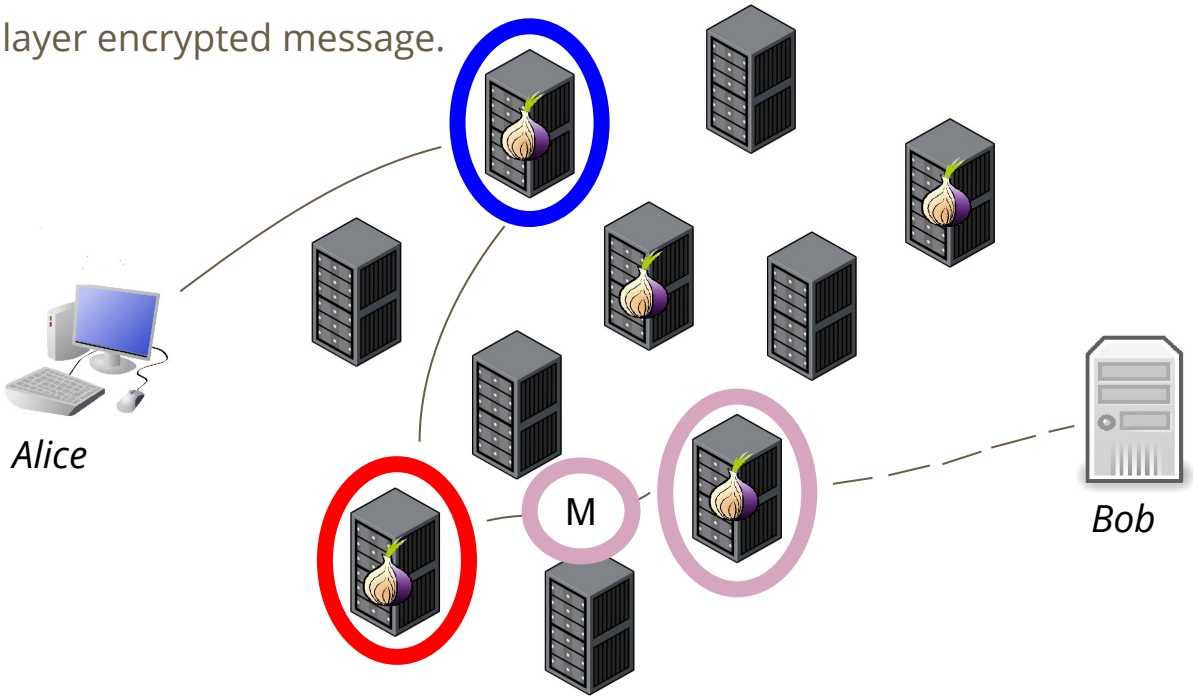
# How Tor works cont.

**Step 4b)** The second node strips the message of one more layer of encryption using the symmetric key that Alice used to encrypt the already 1-layer encrypted message. The second node knows the first node sent the message and now forwards the single layer encrypted message to the third and final node.



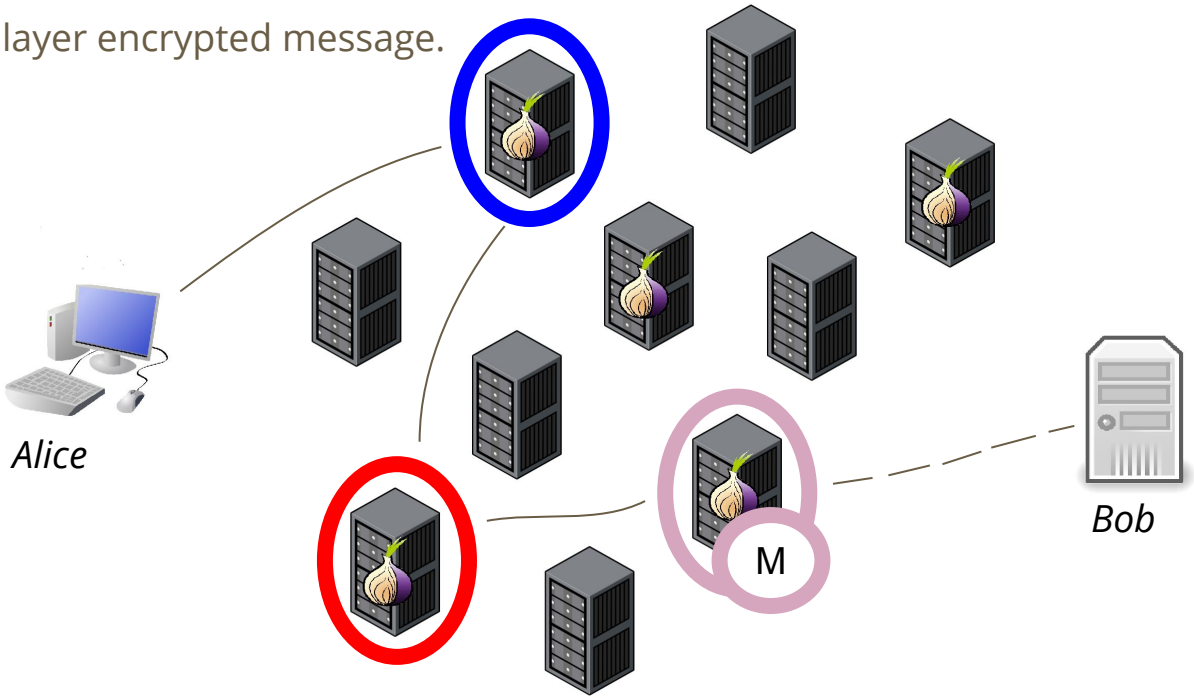
# How Tor works cont.

**Step 5a)** The message travels to the third node who knows the second node sent the single layer encrypted message.



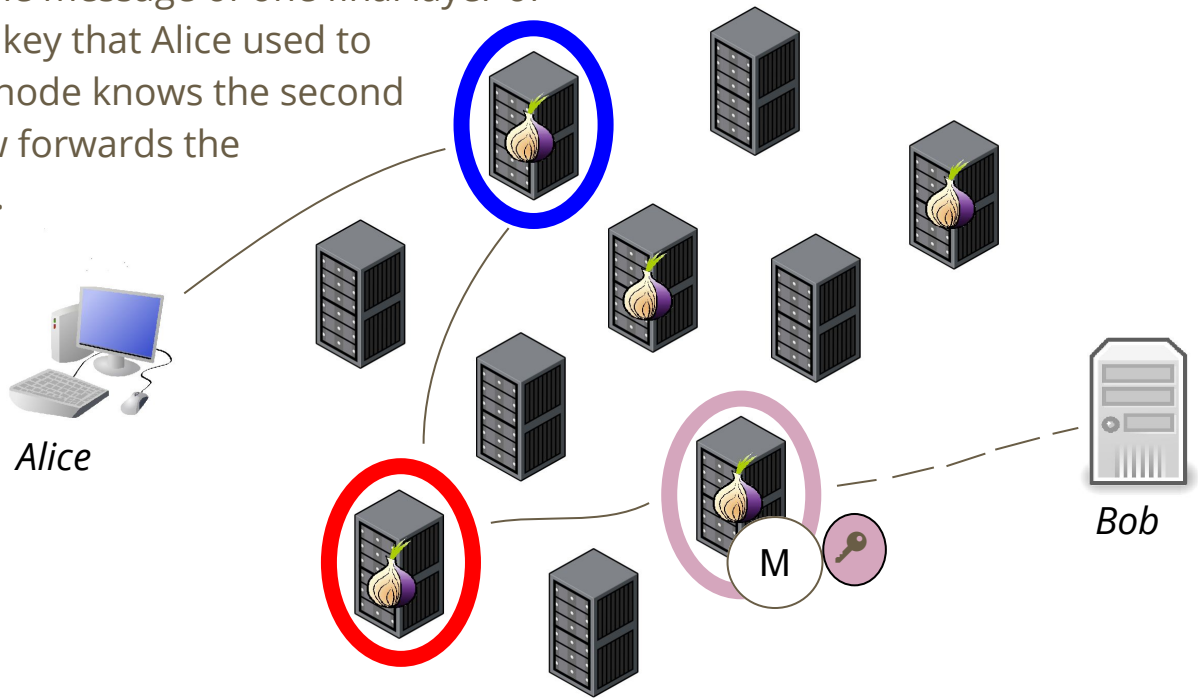
# How Tor works cont.

**Step 5a)** The message travels to the third node who knows the second node sent the single layer encrypted message.



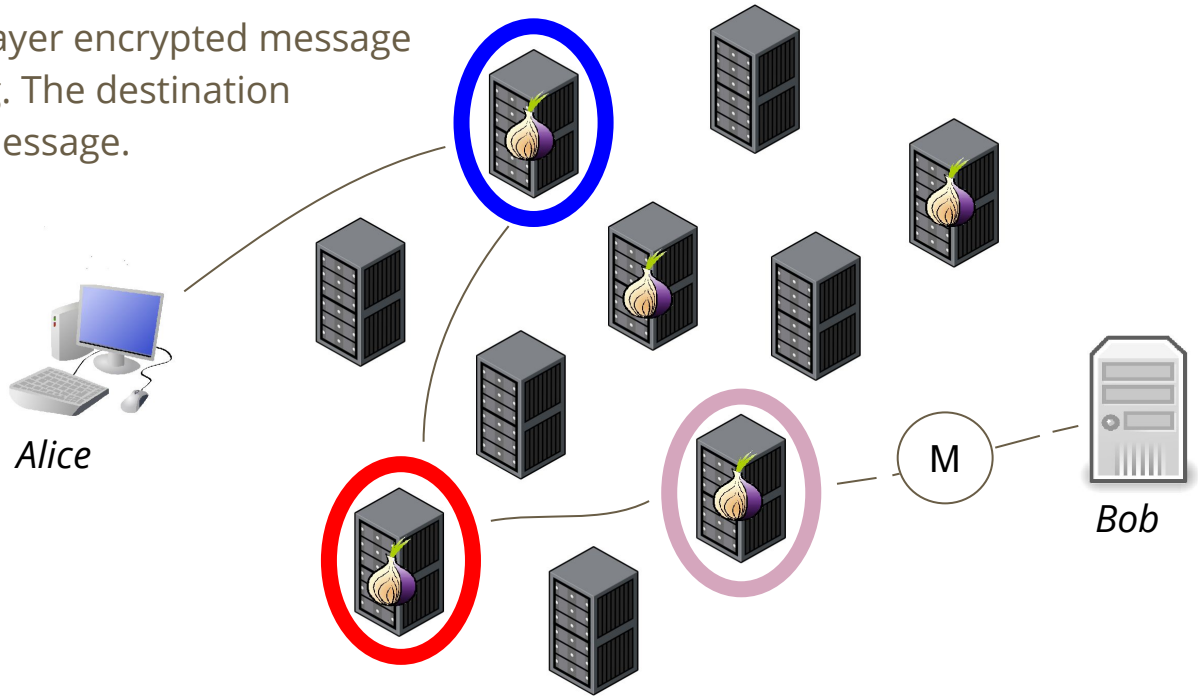
# How Tor works cont.

**Step 5b)** The third node strips the message of one final layer of encryption using the symmetric key that Alice used to encrypt the message. The third node knows the second node sent the message and now forwards the message to the destination Bob.



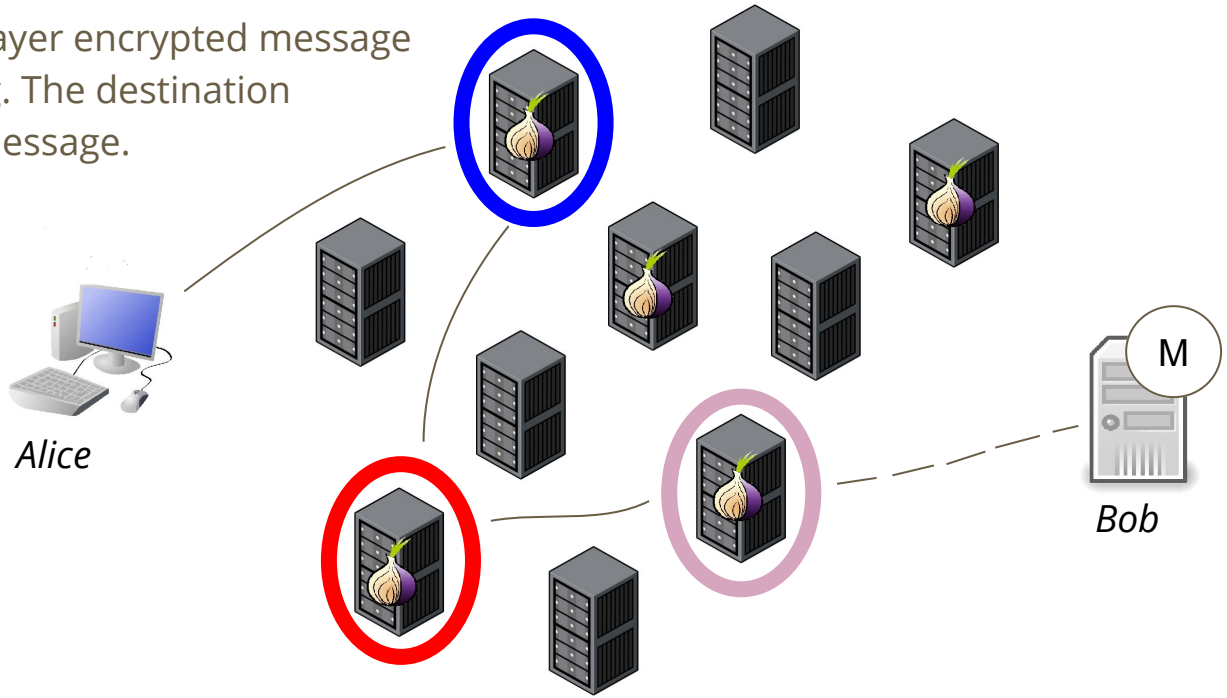
# How Tor works cont.

**Step 6a)** The message travels to the destination. The final node knows who sent the one-layer encrypted message and where the message is going. The destination knows the final node sent the message.



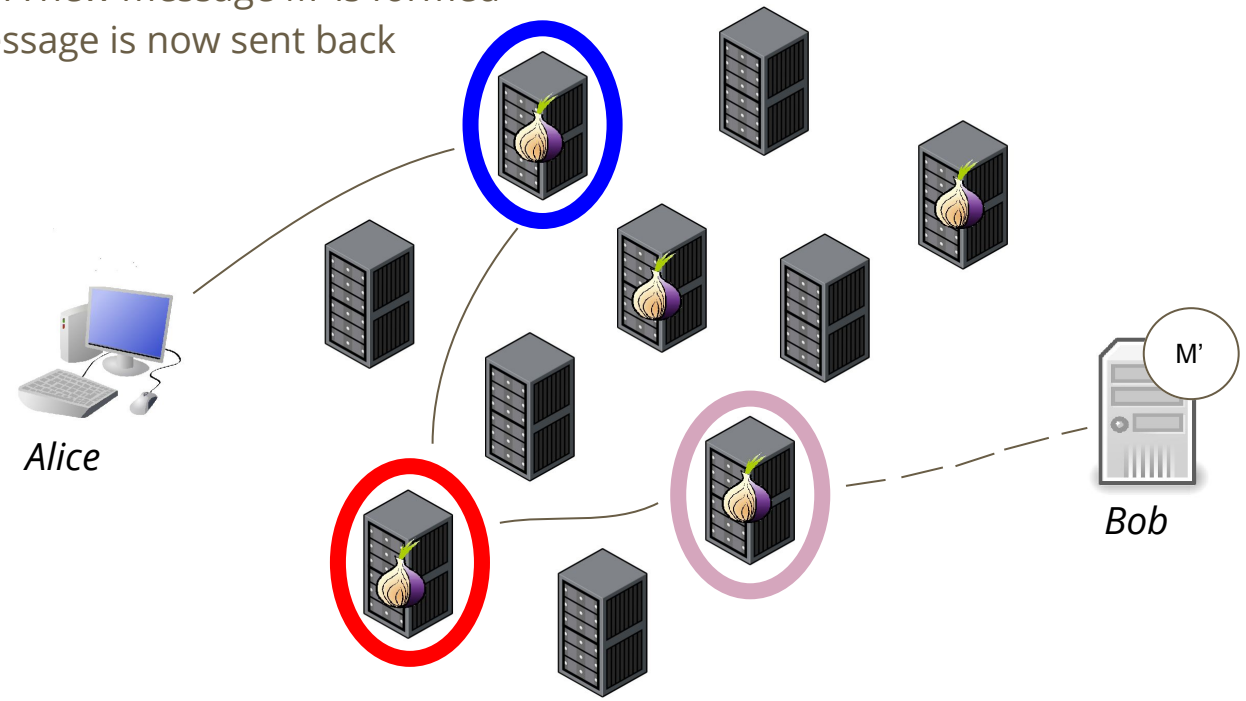
# How Tor works cont.

**Step 6a)** The message travels to the destination. The final node knows who sent the one-layer encrypted message and where the message is going. The destination knows the final node sent the message.



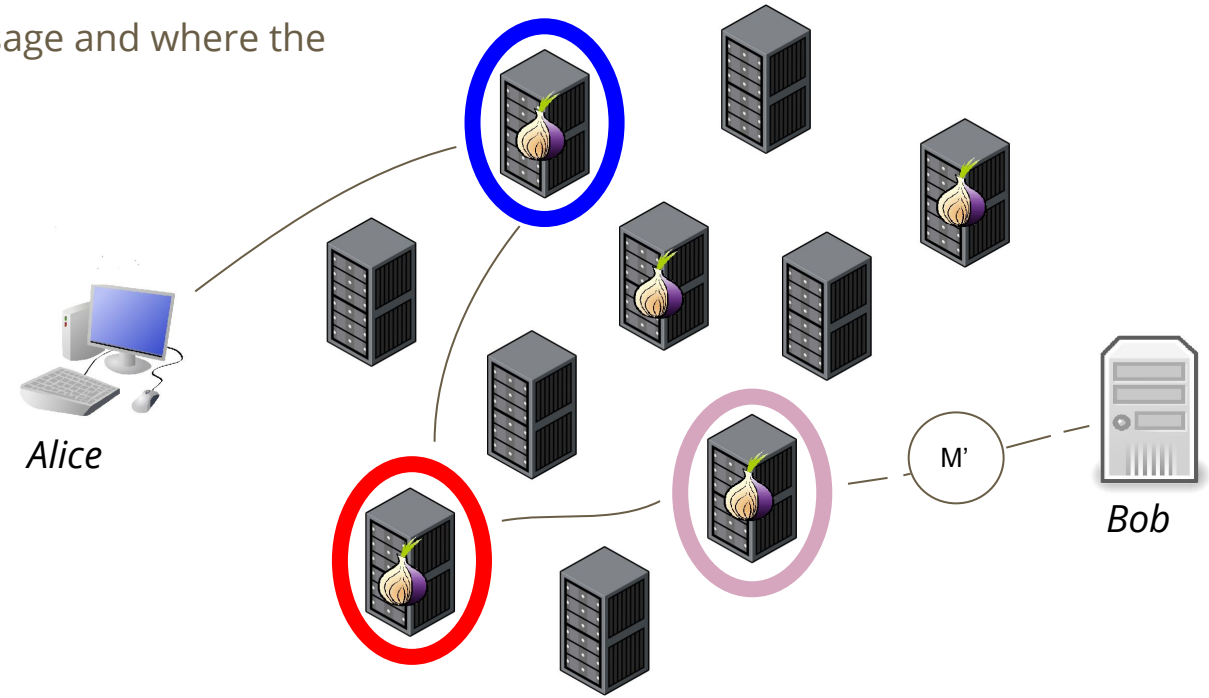
# How Tor works cont.

**Step 7)** The request is handled. A new message M’ is formed by the destination Bob. This message is now sent back through the circuit.



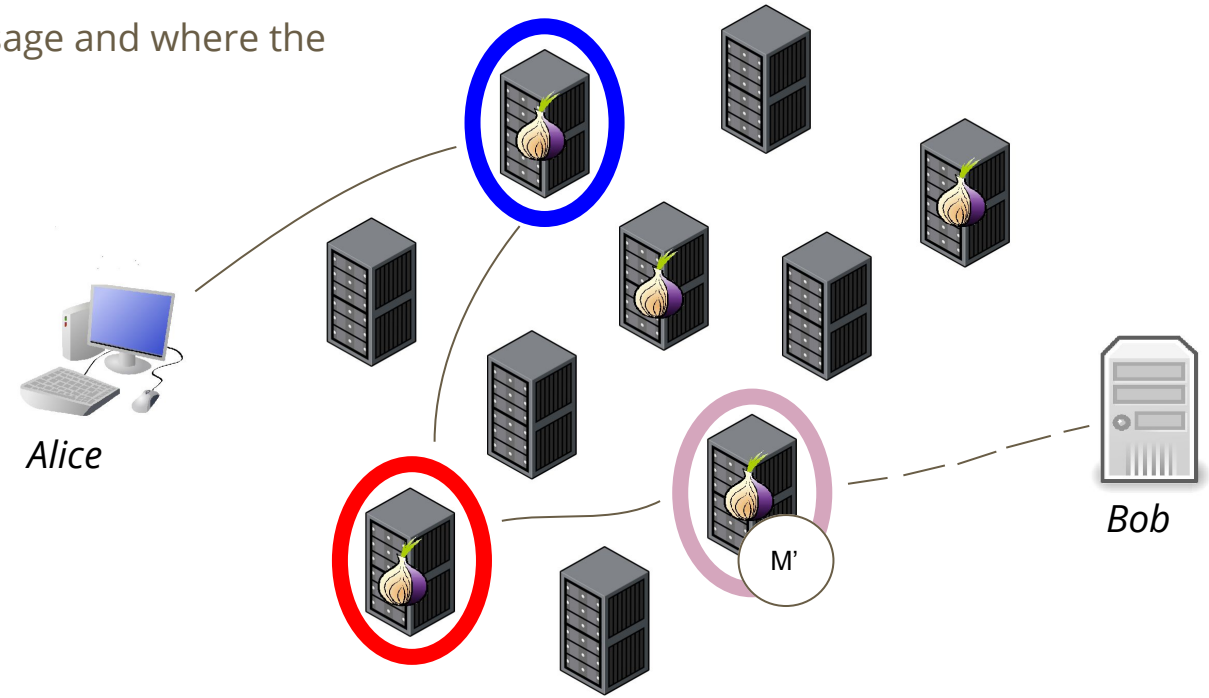
# How Tor works cont.

**Step 8a)** The message travels to the final node. The final node knows who sent the message and where the message is going.



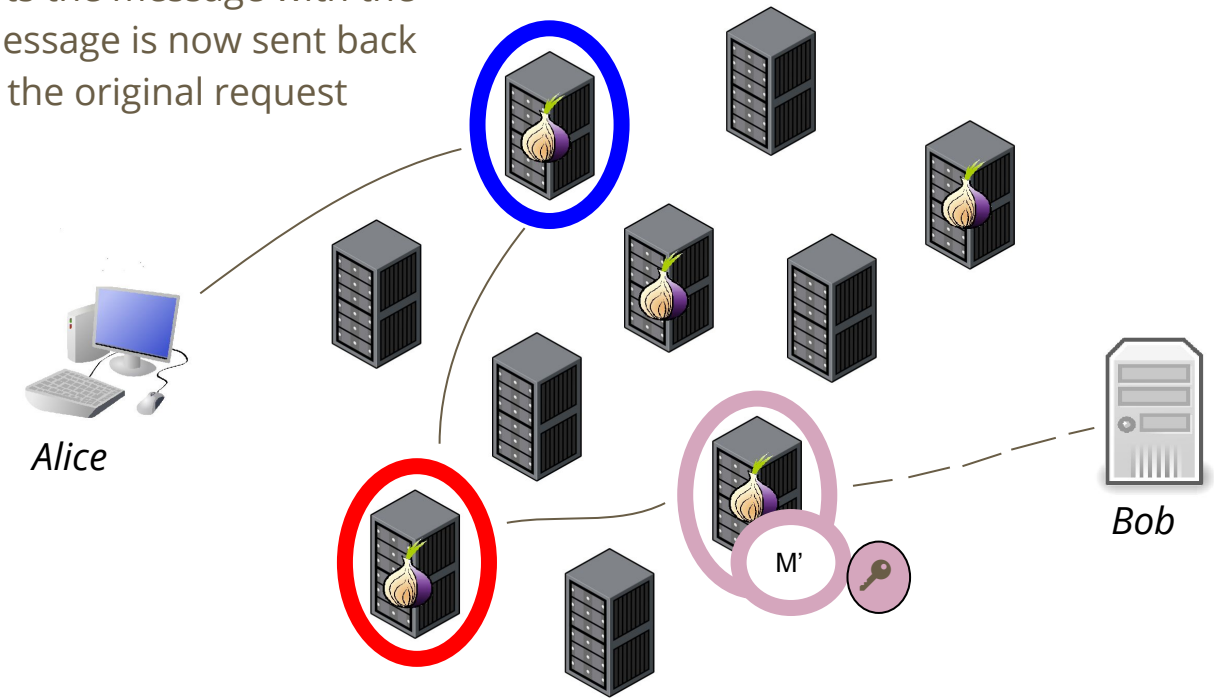
# How Tor works cont.

**Step 8a)** The message travels to the final node. The final node knows who sent the message and where the message is going.



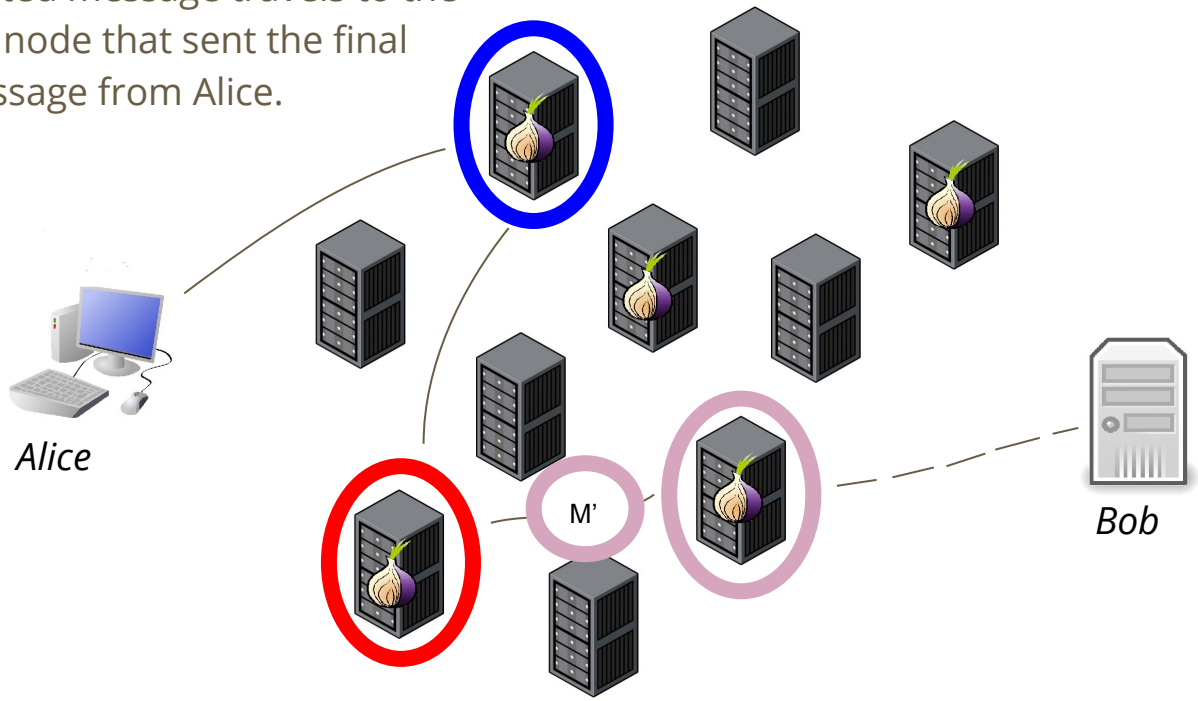
# How Tor works cont.

**Step 8b)** The third node encrypts the message with the symmetric key Alice has. The message is now sent back to the node that sent this node the original request from Alice.



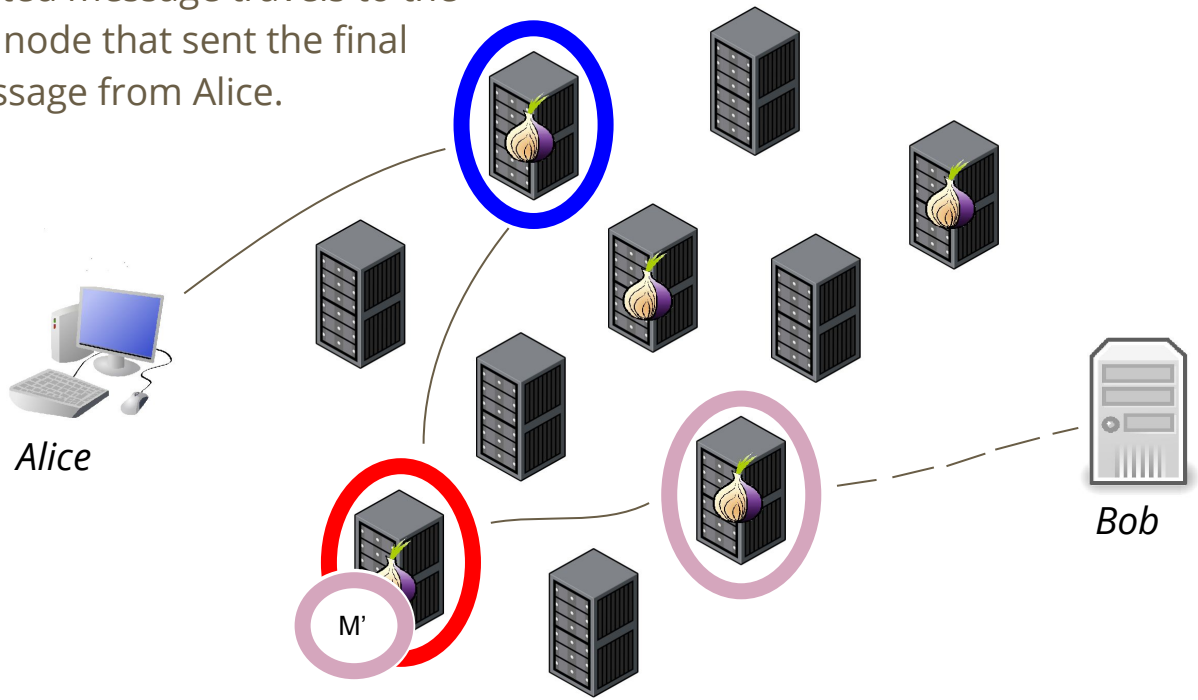
# How Tor works cont.

**Step 9a)** The single layer encrypted message travels to the middle node because this is the node that sent the final node the original encrypted message from Alice.



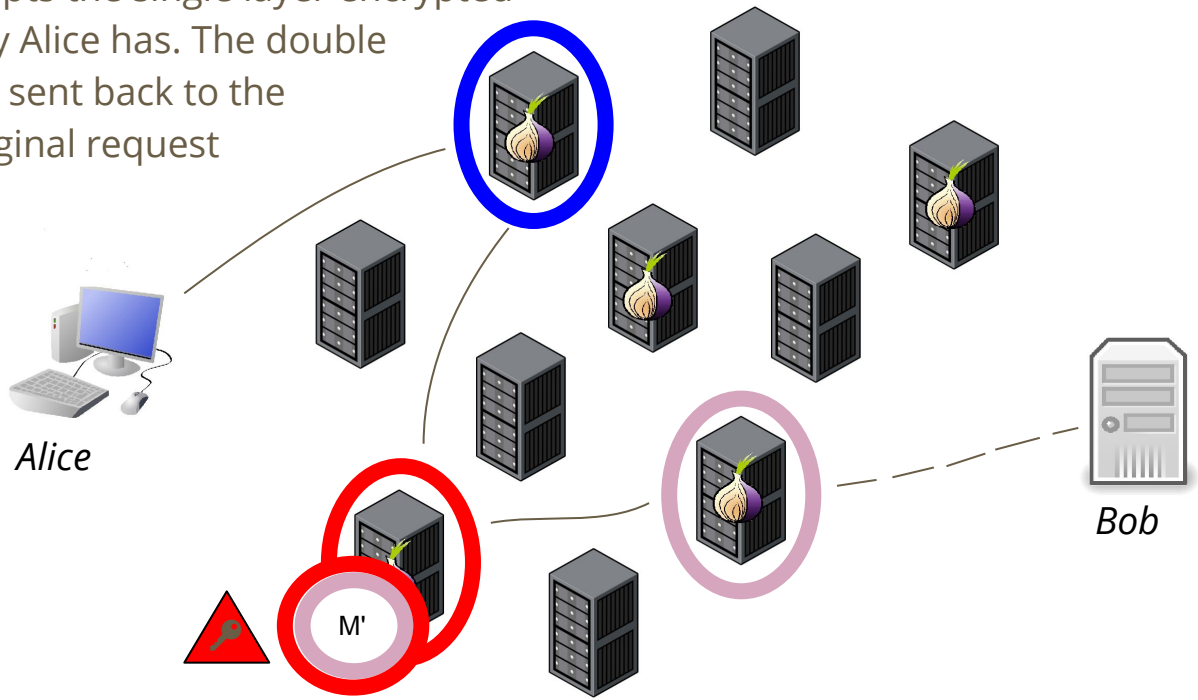
# How Tor works cont.

**Step 9a)** The single layer encrypted message travels to the middle node because this is the node that sent the final node the original encrypted message from Alice.



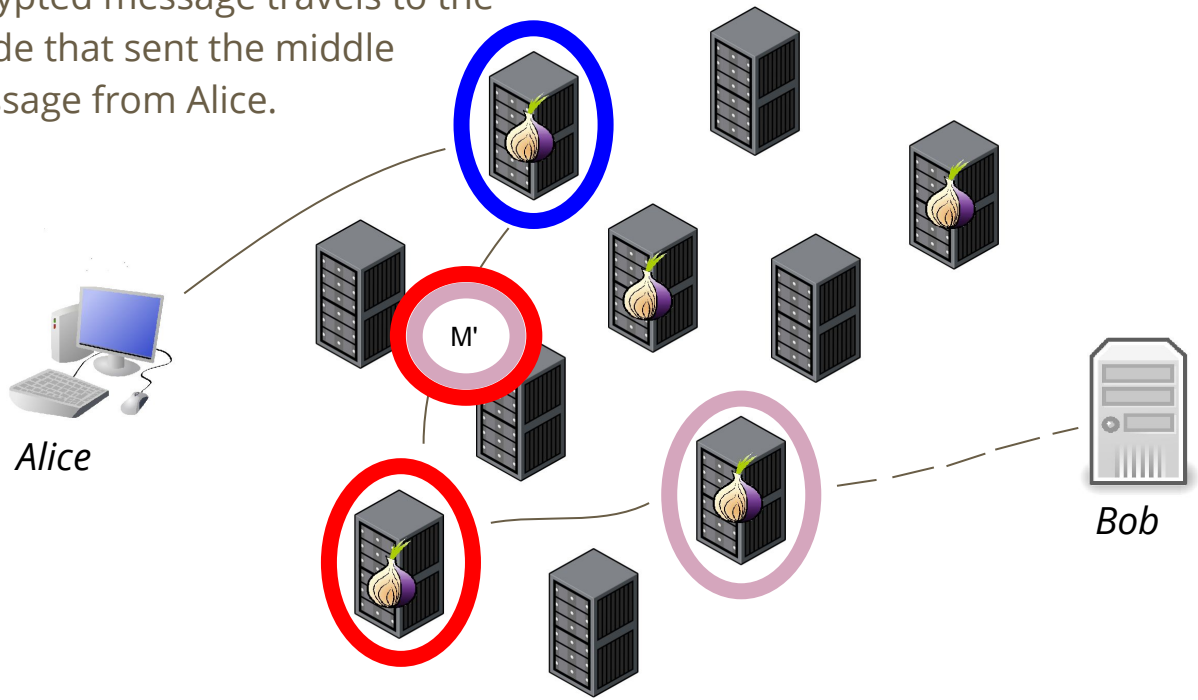
# How Tor works cont.

**Step 9b)** The middle node encrypts the single layer encrypted message with the symmetric key Alice has. The double layer encrypted message is now sent back to the node that sent this node the original request from Alice.



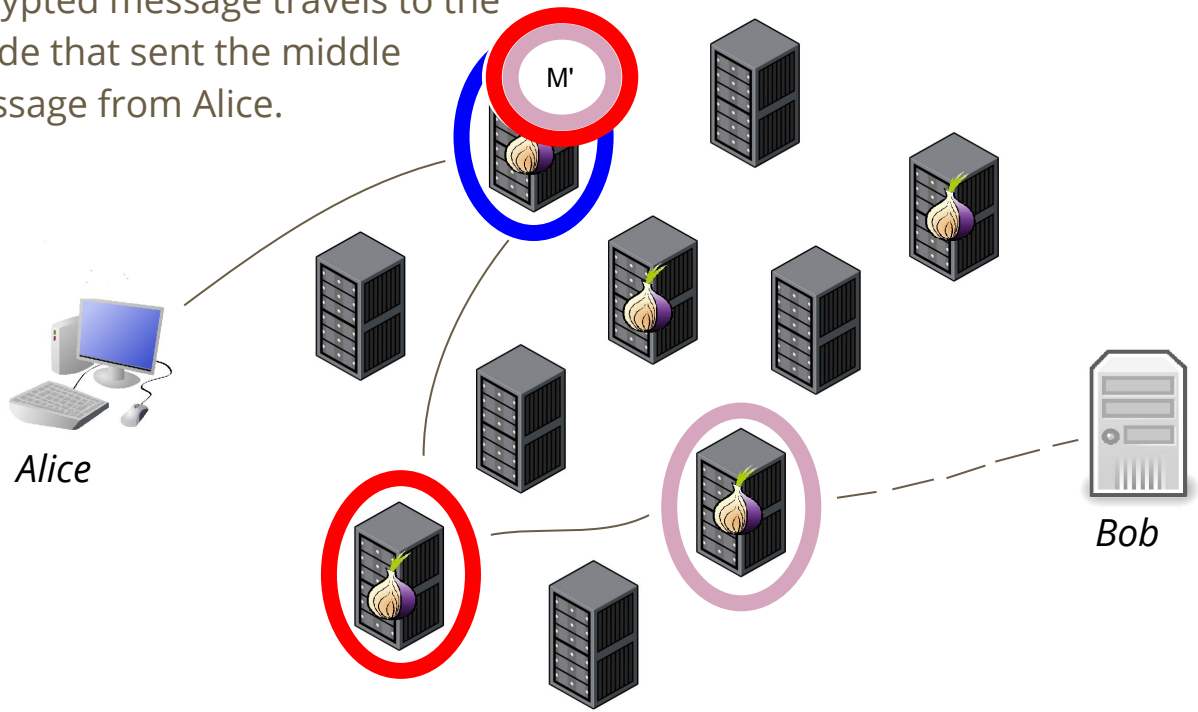
# How Tor works cont.

**Step 10a)** The double layer encrypted message travels to the first node because this is the node that sent the middle node the original encrypted message from Alice.



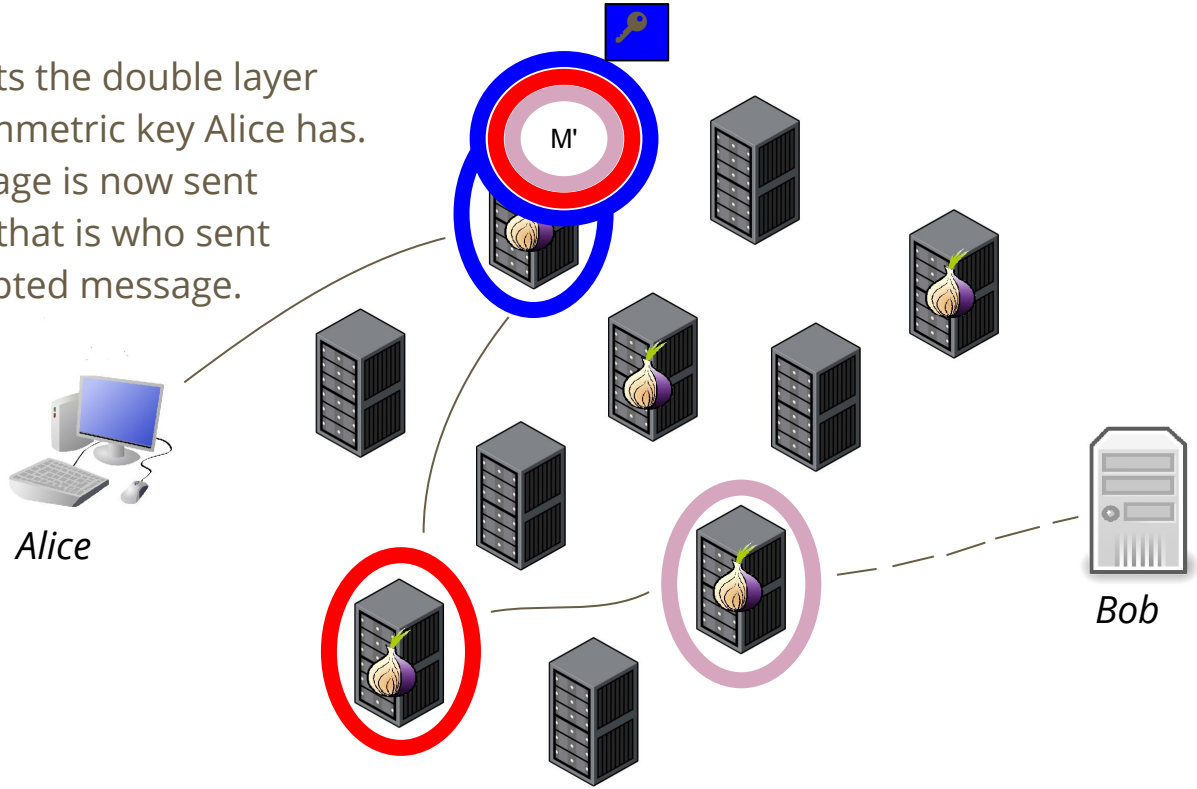
# How Tor works cont.

**Step 10a)** The double layer encrypted message travels to the first node because this is the node that sent the middle node the original encrypted message from Alice.



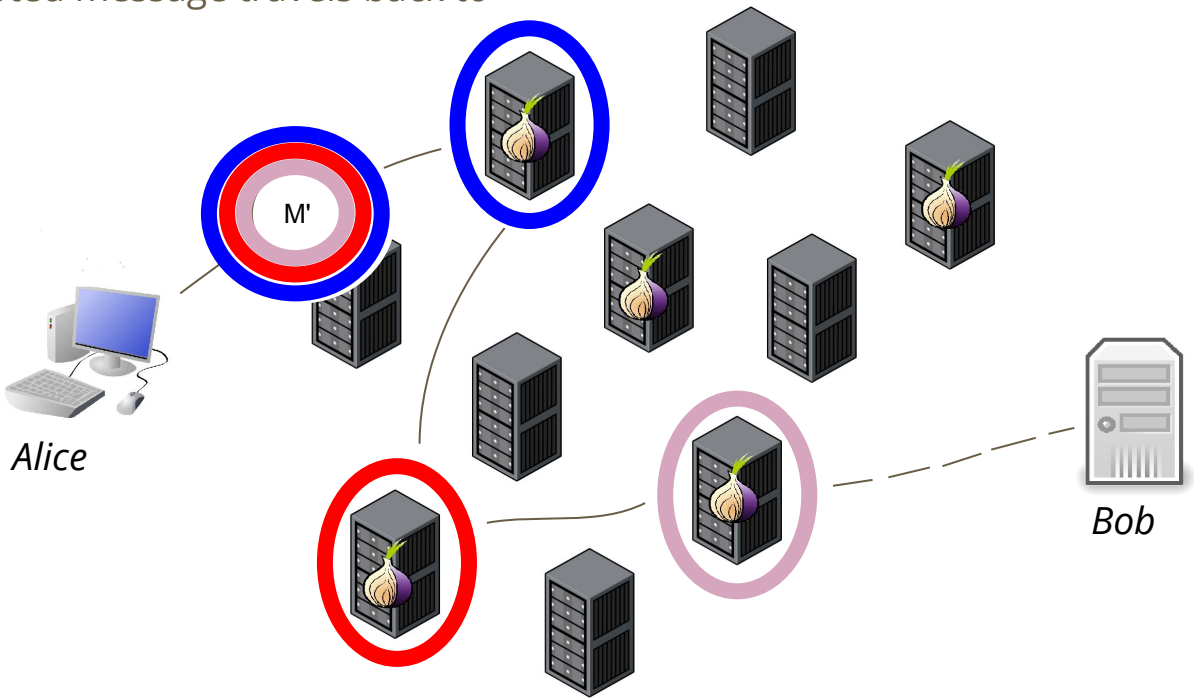
# How Tor works cont.

**Step 10b)** The first node encrypts the double layer encrypted message with the symmetric key Alice has. The triple layer encrypted message is now sent back to the client Alice because that is who sent the first node the original encrypted message.



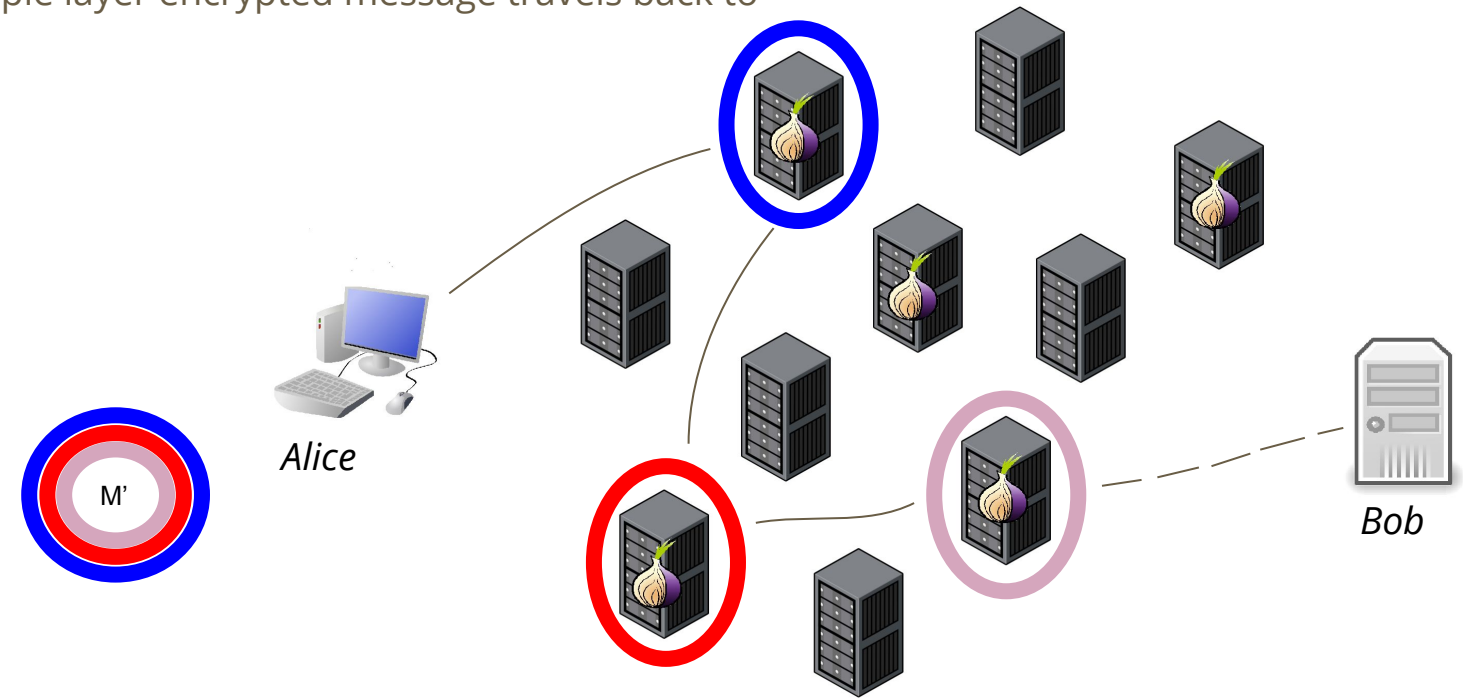
# How Tor works cont.

**Step 11a)** The triple layer encrypted message travels back to Alice.



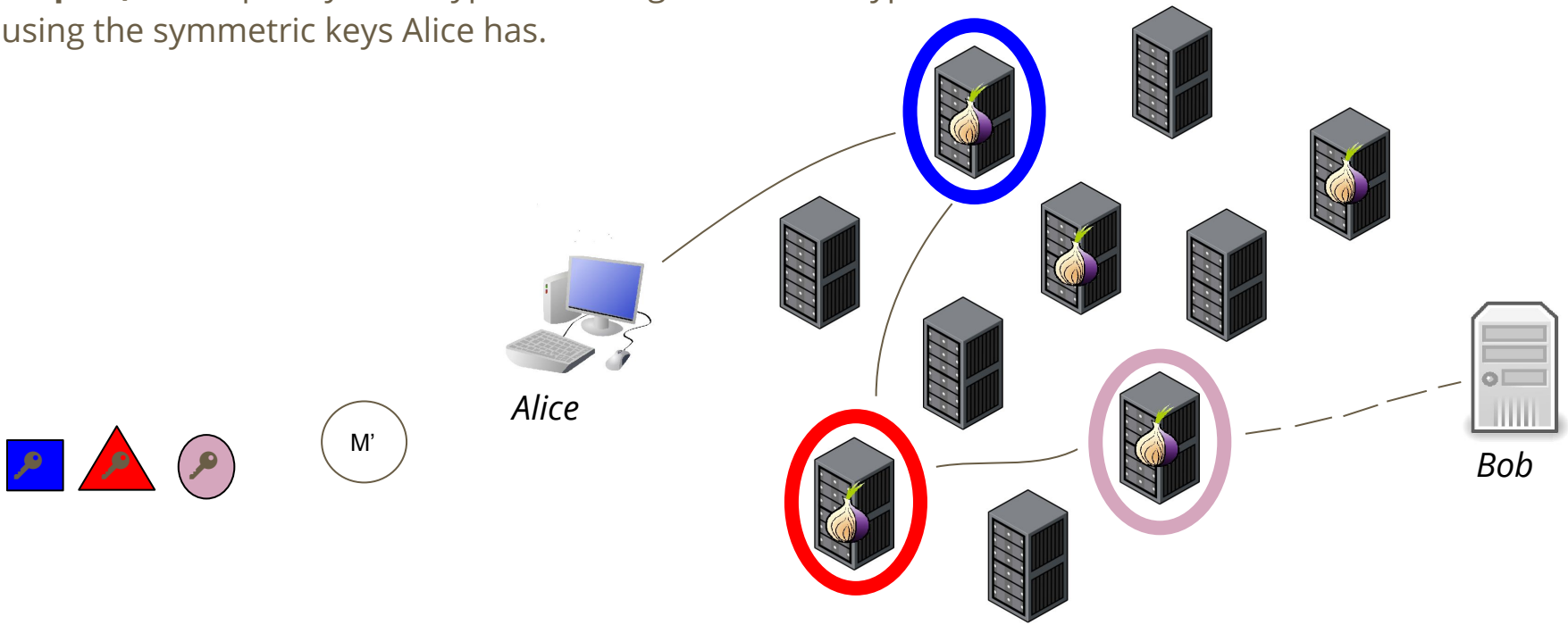
# How Tor works cont.

**Step 11a)** The triple layer encrypted message travels back to Alice.



# How Tor works cont.

**Step 12)** The triple layer encrypted message is now decrypted using the symmetric keys Alice has.



# Tor and HTTPS

- Tor
  - Hides who you are and where you’re going
- HTTPS
  - Hides the information you’re sending
- Tor and HTTPS
  - Hides who you are, where you’re going and the information you’re sending

Visualised: https://www.eff.org/pages/tor-and-https

# The Tor Browser

- A Firefox Browser preconfigured to connect to the Tor network
  - Comes with altered Firefox settings, ad-ons such as HTTPS Everywhere and NoScript
- Most secure way to using Tor, why?
  - Before getting into Tor network -  Have done all the preparations (includes proxy or VPN)
  - While browsing - refreshing identity manually or automatically, no history
  - Leaving Tor network - deleting all the cookies, caches, and any other information that relates to your privacy when you were browsing
- Project Fusion - integrated Tor into Firefox

# Using Tor through Terminal

- Curl
  - Basically works like a proxy
  - curl -s --socks5-hostname [Tor HOST:PORT] [URL]
  - curl -s --socks5-hostname localhost:9050 ifconfig.me

- Torify
  - Torify [any command you want to use]
  - Torify wget [some URL]

# How Tor works - Services

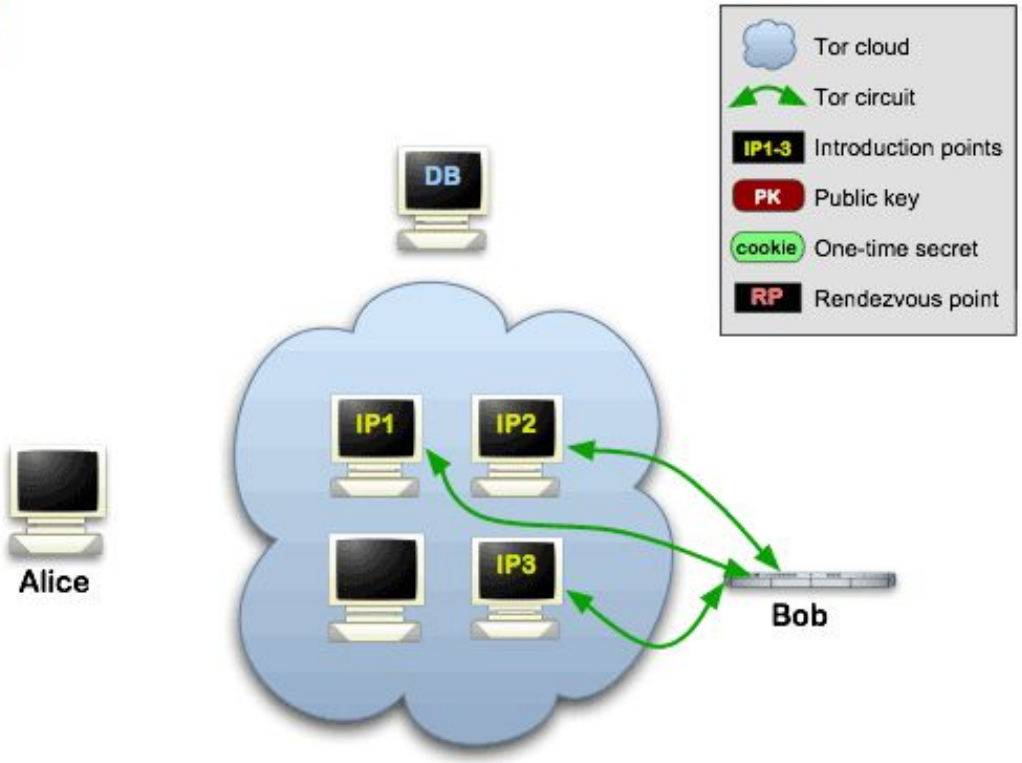
- A user can connect to onion services or clearnet services
- Clearnet services
  - E.g. wikipedia.org, facebook.com
  - Some have onion service equivalents (e.g. facebook, thepiratebay etc.)
- Onion services
  - E.g. duckduckgos onion service - https://3g2upl4pq6kufc4m.onion/ Facebooks onion service - https://facebookcorewwwi.onion/
  - In the Tor network, whoever runs the service remains anonymous
  - Keep Tor services and their providers anonymous to clients and vise versa

# Onion Services

Bob has a website running on some server and wishes to keep his location unknown to others.

...

Bob can set up an onion service!



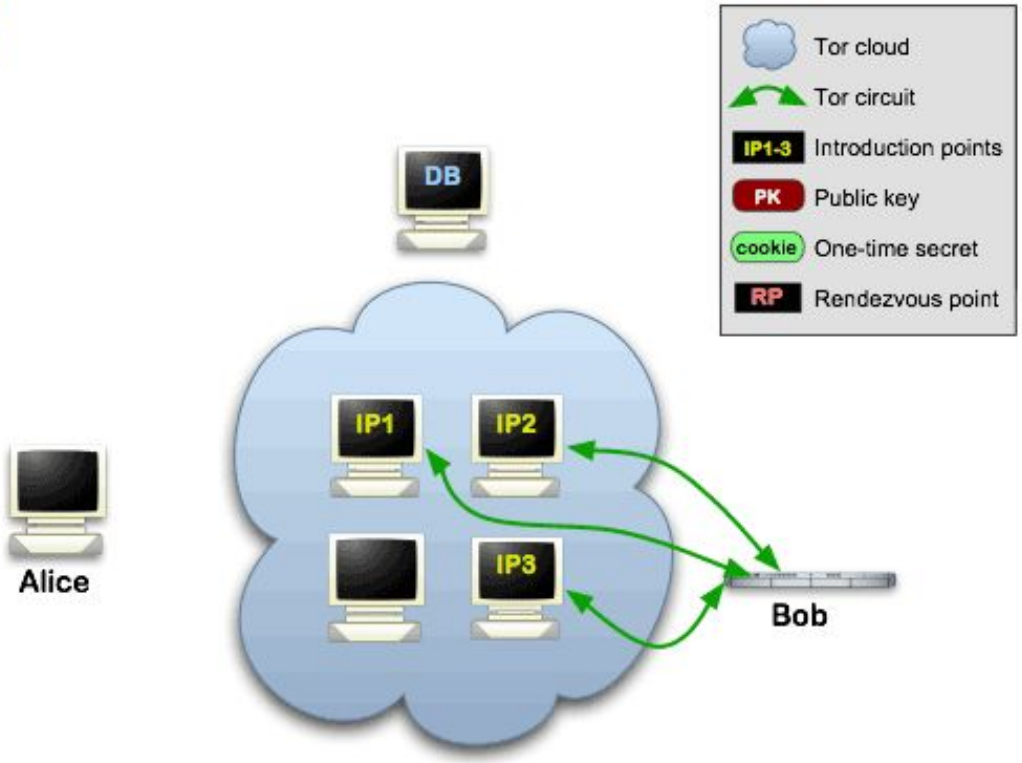
# Onion Services

Bob sets up 3 circuits, each circuit goes to an *introduction point.* Introduction points are relays chosen randomly by the onion service.

**On the right:**

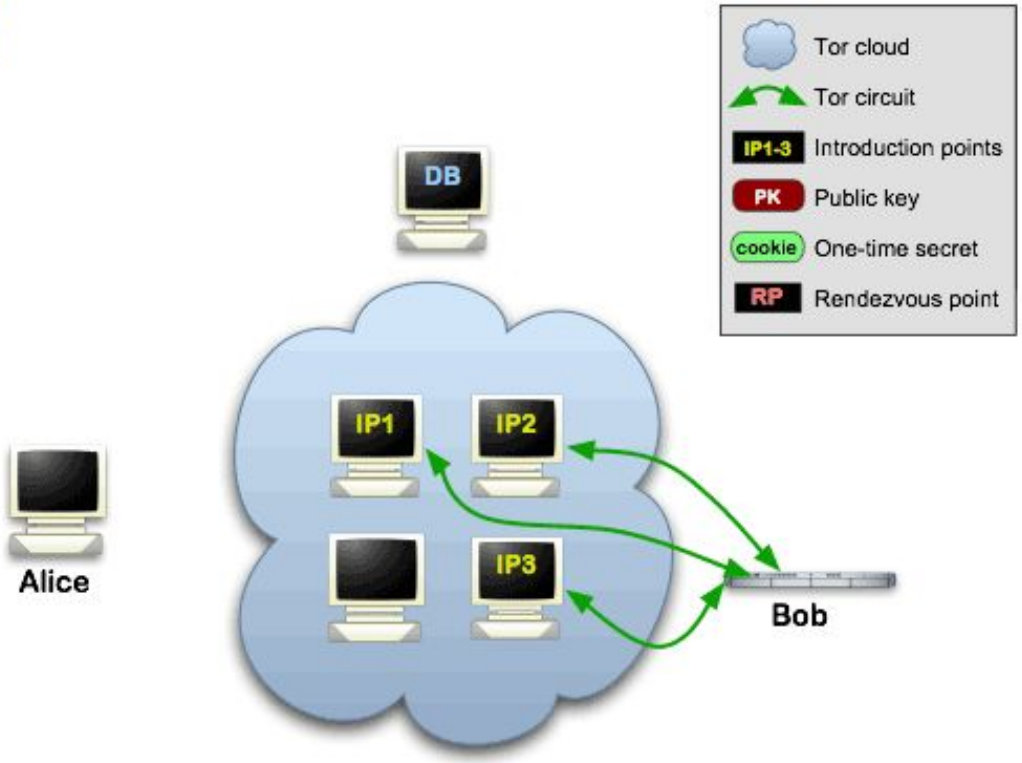
IP1-3 are introduction points

Arrows from Bobs server to the introduction points are **Tor circuits.** The introduction points don't know where Bob is. Noone can truly find out where Bob's real location is.



# Onion Services

Bob sends these introduction points his public key so that they (and other people) will know that this is Bobs service. They just won't know where Bob is located.



# Onion Services

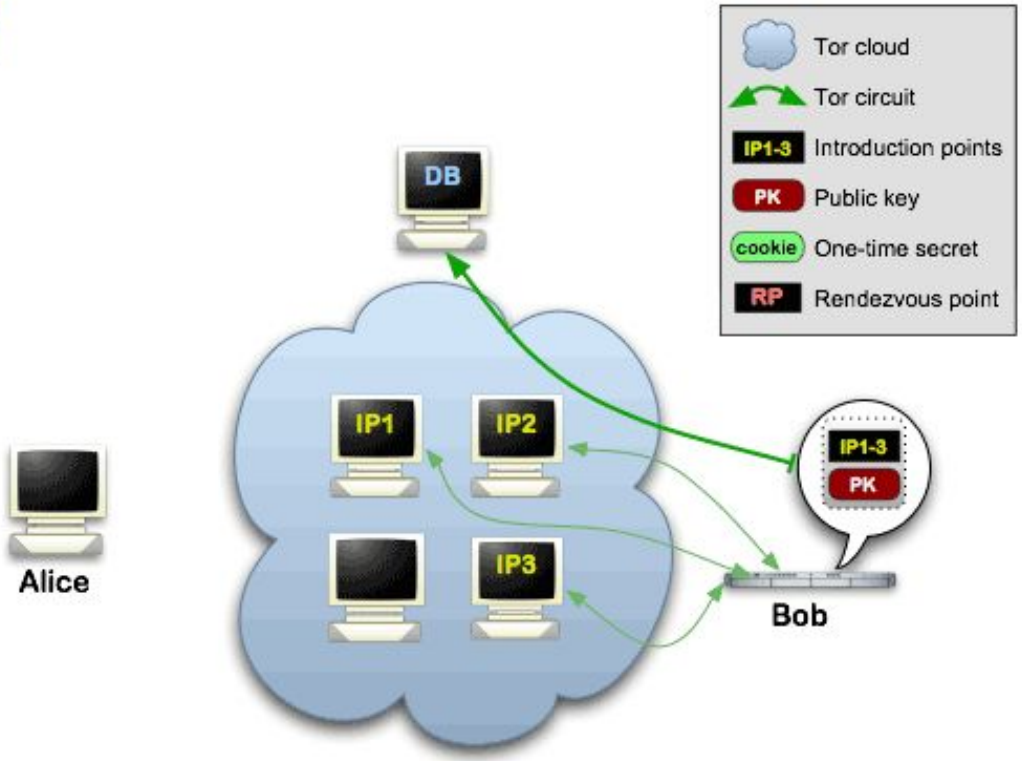
The service now makes a *onion service descriptor*.

This *onion service descriptor* contains:

- Public key of Bobs onion service
- Summary of introduction points

This descriptor is signed by the onion services private key.

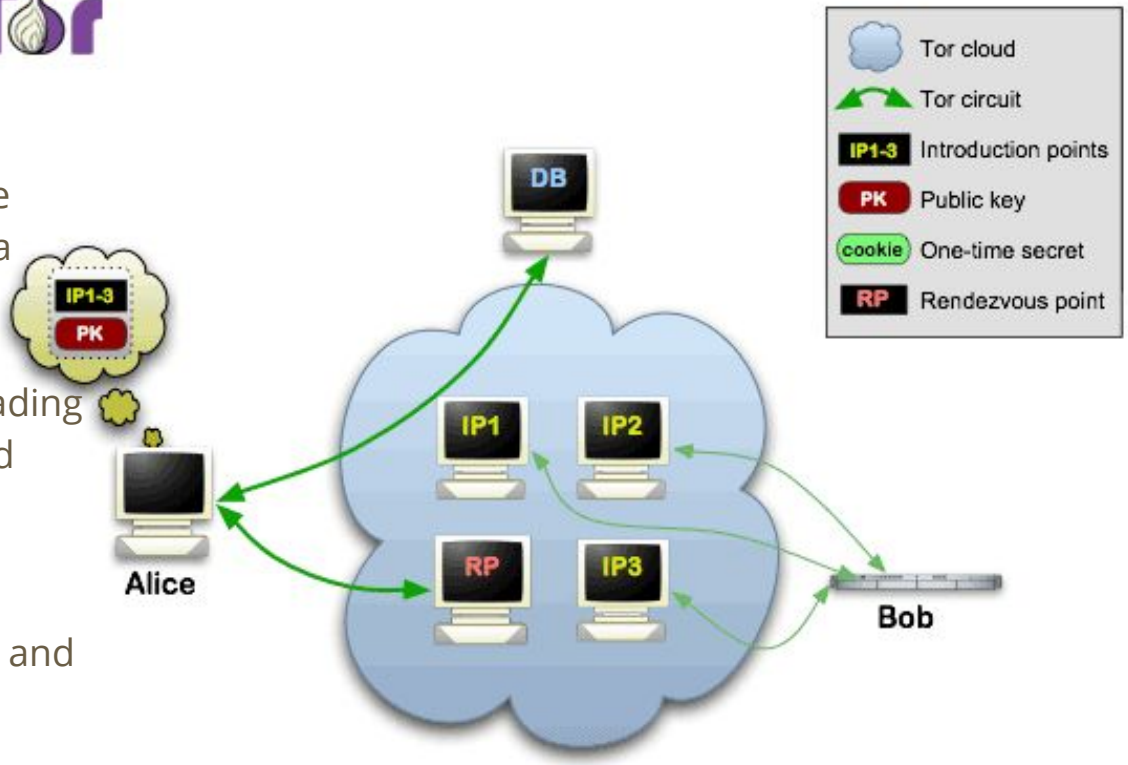
This information is uploaded to DB (on the right). DB is a distributed hash table where the address of the onion service corresponds to the descriptor.



# Onion Services

The client Alice knows the onion address of the onion service. (Alice could have find out about it from a friend or may have seen it online somewhere.) Alice will begin to establish a connection by downloading the descriptor from the distributed hash table.

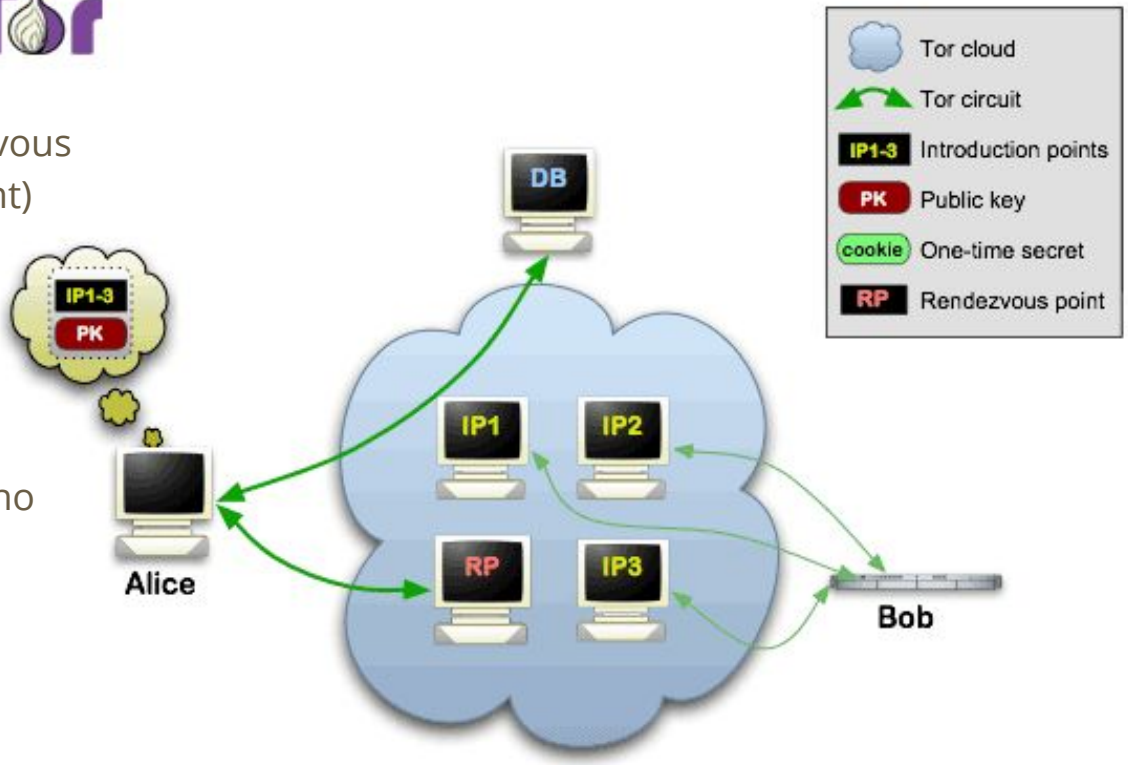
The descriptor tells Alice the introduction points for the service and the public key to use.



# Onion Services

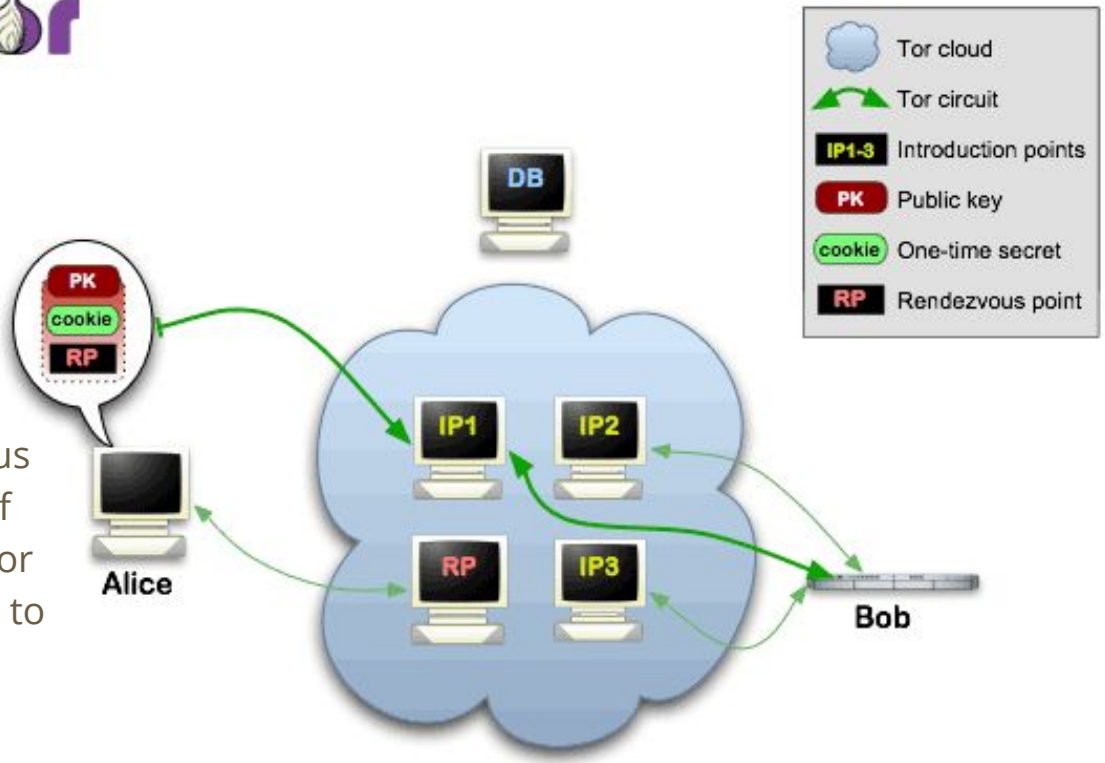
Alice will also connect to a rendezvous point (machine with RP on the right) and sends it a one-time secret.

Alices connection to RP is through a Tor circuit where the last node is the rendezvous point. The rendezvous point doesn't know who Alice is or where she is.



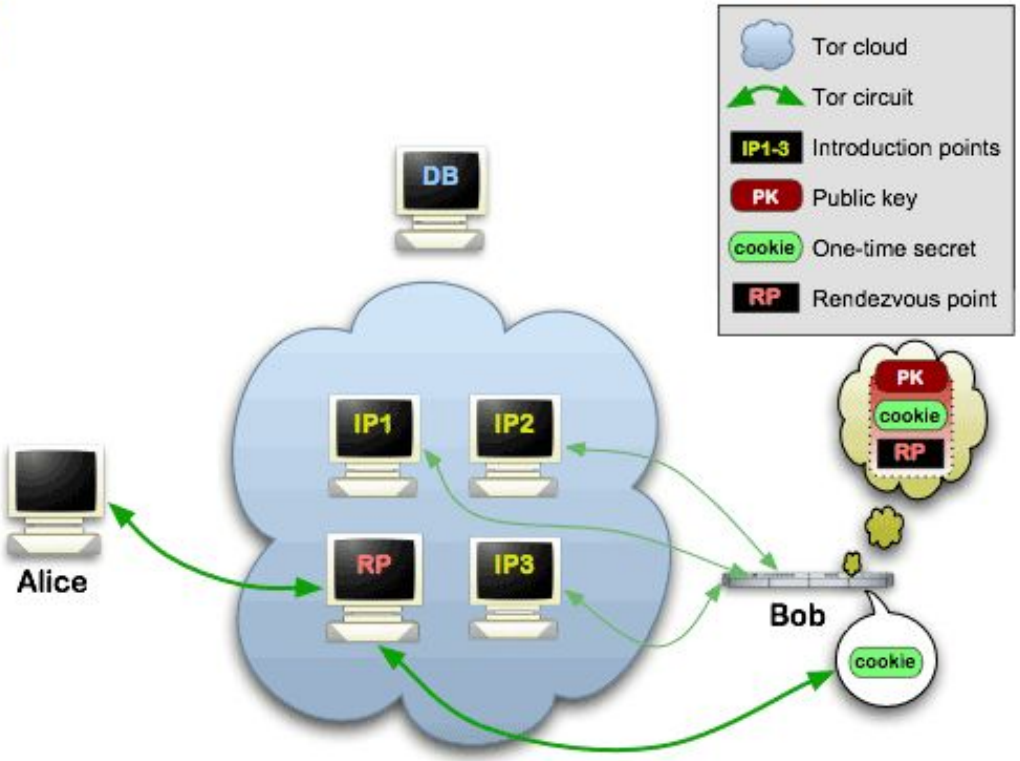
# Onion Services

The client Alice now forms a introduction message which will be encrypted with the onion services public key. The message contains the address of the rendezvous point and the one-time secret that Alice has shared with the rendezvous point. The message is sent to one of the introduction points through a Tor circuit and is going to be forwarded to the onion service.



# Onion Services

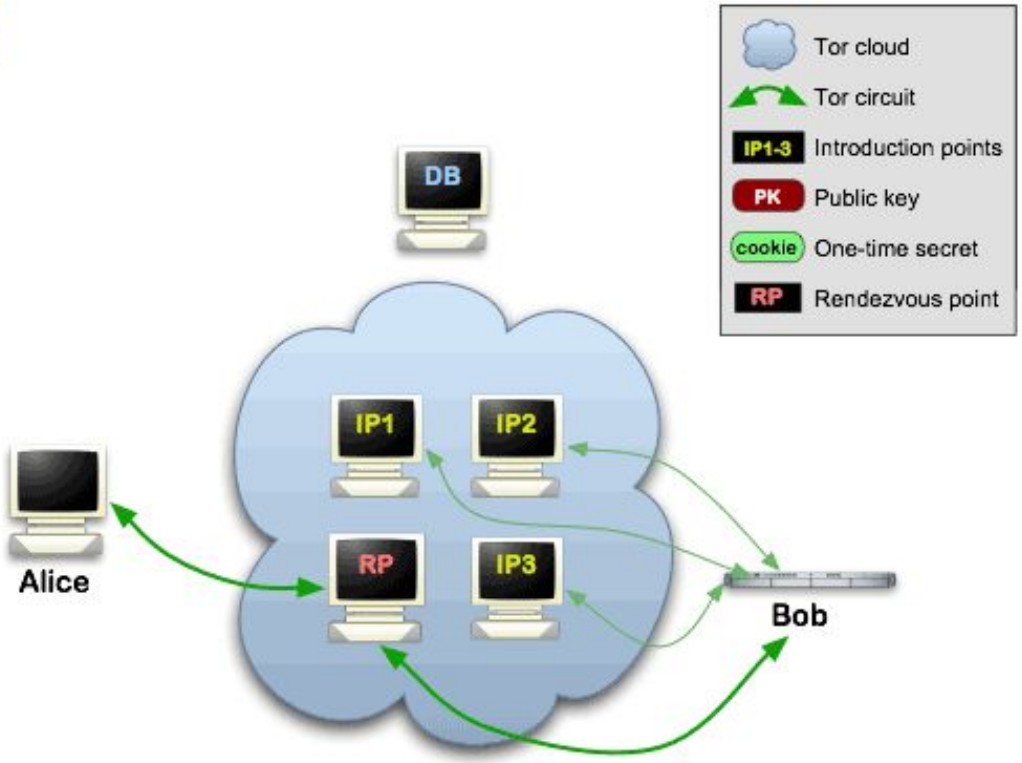
The onion service decrypts the message. The onion service will now create a **new Tor circuit** to the rendezvous point with a message that contains the one-time secret that only Alice has shared with the rendezvous point.



# Onion Services

The rendezvous point now notifies the client the connection to the onion service has been established.

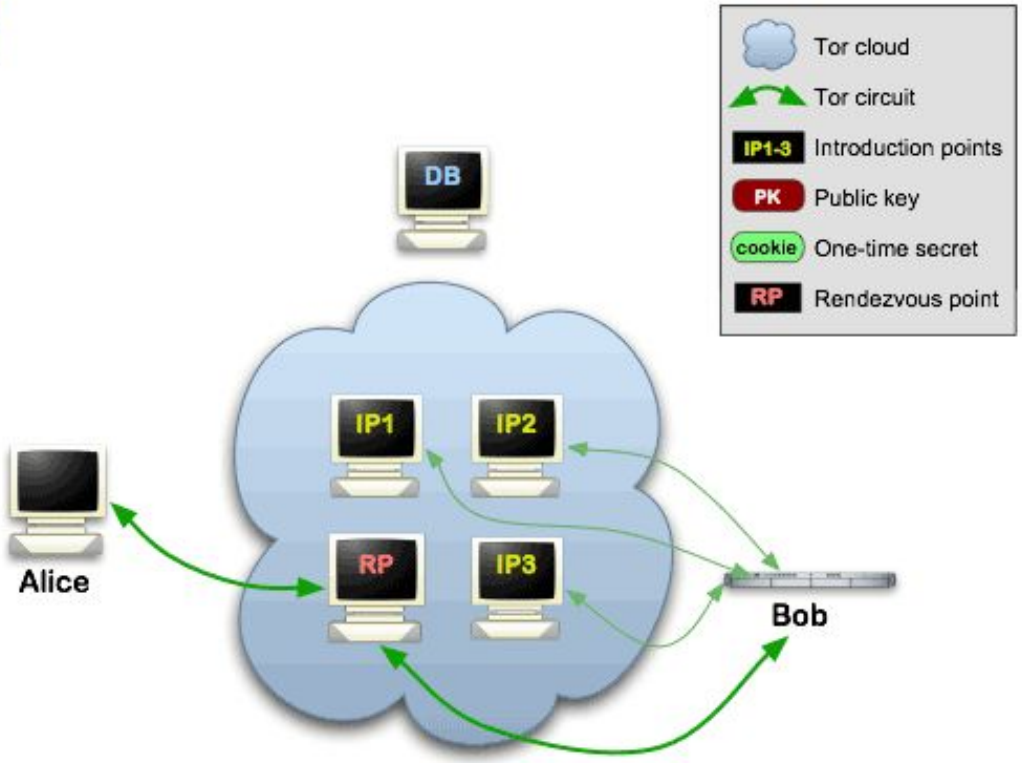
The rendezvous point just allows the client and onion service to communicate with each other. This entire circuit is 6 nodes total, 3 (including RP) for Alice and 3 for the onion service. Fully encrypted.



# Onion Services

Bob doesn’t know who or where Alice is.

Alice doesn’t know where Bob is.



Tor Onion Services images and in-depth explanation:

https://2019.www.torproject.org/docs/onion-services.html.en

# Tor vs. VPN

VPN

- Protects your privacy by encrypting your message and hide your IP address
- Relies on VPN Provider - you have to have a trusty VPN  service to Actually to protect your privacy( your won't be anonymous when VPN Provider get compromise)

Tor

- Communicate anonymously
- No organization rcontrol
- No worries about trusting any organization to use.

# What Not to Do with Tor

- Torrent over Tor
- Enable or install plugins
- Using non-HTTPS version of website
- Don’t open docs downloaded through Tor while online

# References


https://2019.www.torproject.org/download/download.html.en#Warning

https://www.csoonline.com/article/3287653/what-is-the-tor-browser-how-it-works-and-how-it-can-help-you-protect-your-identity-online.html

https://thebestvpn.com/tor-vs-vpn/

https://hackernoon.com/how-does-tor-really-work-5909b9bd232c

https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf

https://2019.www.torproject.org/docs/onion-services.html.en

https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt