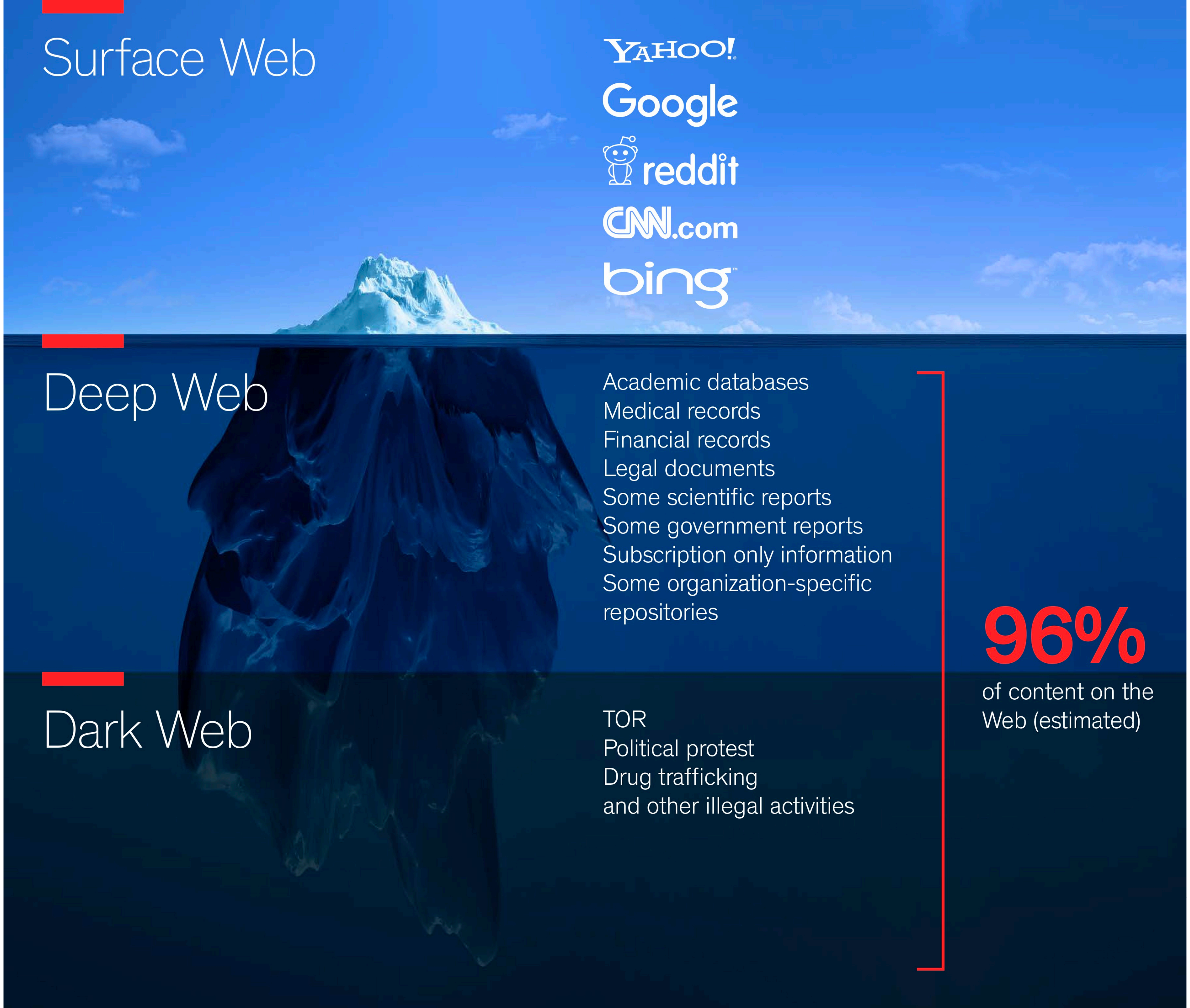
Lesson 07-02:
Network Security - Tor Hidden Service

CS 356 Computer Networks

Mikyung Han

mhan@cs.utexas.edu

Tor: Enabling Anonymous Communication Over the Internet



Surface Web

YAHOO!
Google
reddit
CNN.com
bing

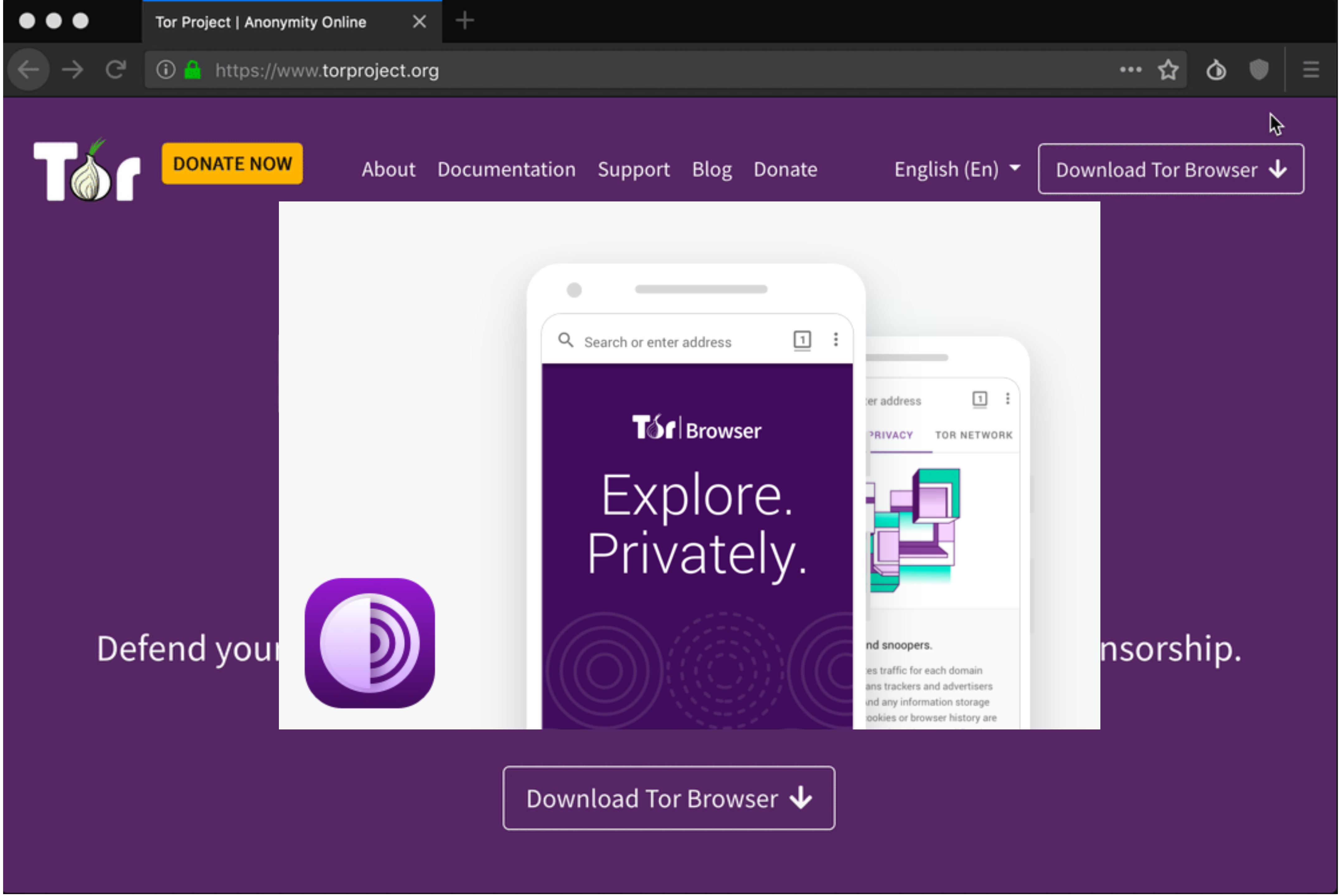
Deep Web

Academic databases
Medical records
Financial records
Legal documents
Some scientific reports
Some government reports
Subscription only information
Some organization-specific repositories

Dark Web

TOR
Political protest
Drug trafficking
and other illegal activities

96%
of content on the Web (estimated)



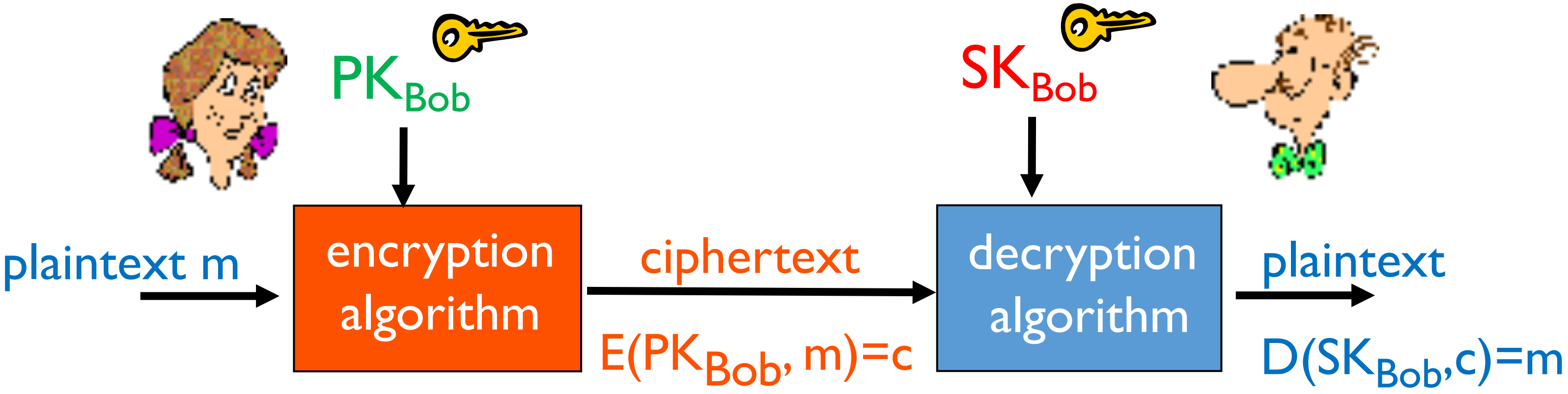
Defend you nsorship.

nd snoopers.
es traffic for each domain
ans trackers and advertisers
nd any information storage
ookies or browser history are

Outline

1. Network Security Recap

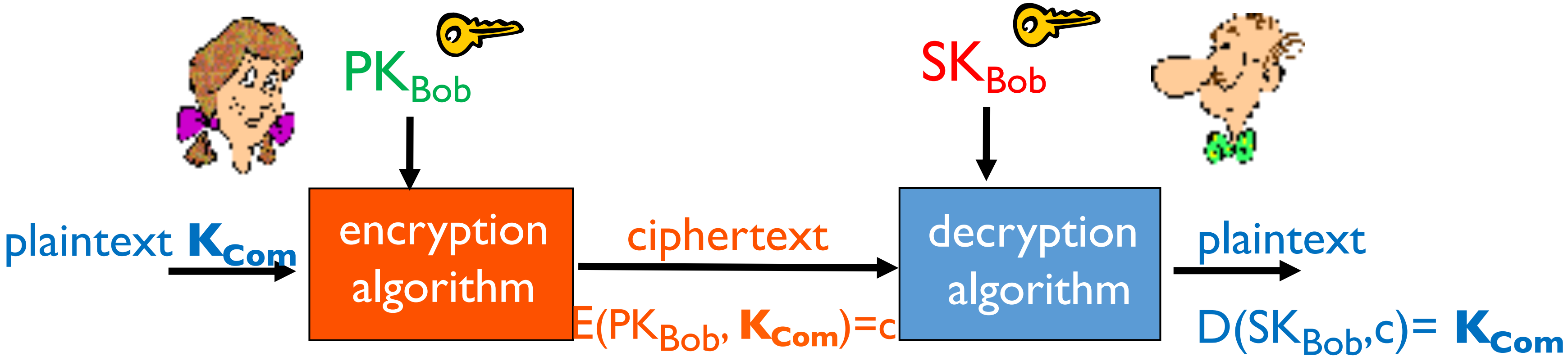
Public Key Infrastructure (PKI)



ex) RSA, Elliptic Curve, etc.

PK public key
SK private key

Alice can send the suggested share key to Bob encrypting with Bob's public key

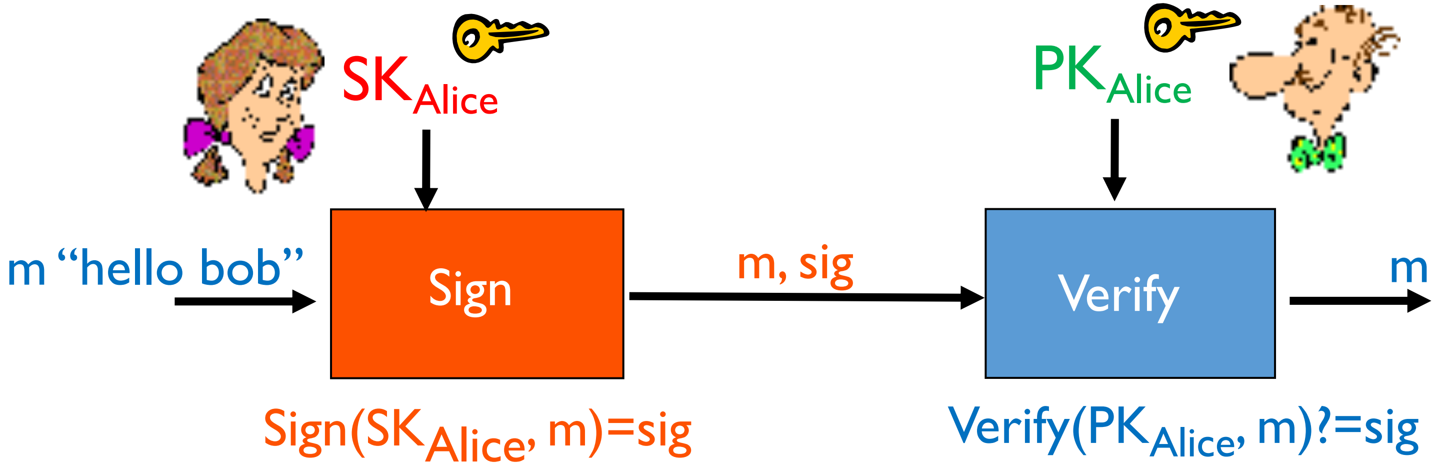

PK_{Bob}
SK_{Bob}
plaintext $\mathbf{K_{Com}}$
encryption algorithm
ciphertext
$E(PK_{Bob}, \mathbf{K_{Com}})=c$
decryption algorithm
plaintext
$D(SK_{Bob},c)= \mathbf{K_{Com}}$

PK public key
SK private key

PKI is also used in digital signature

Provides authenticity and integrity of digital messages

- Authenticity: The message was created by the known sender
- Integrity:The message was not altered in transit



SK_{Alice}

PK_{Alice}

m “hello bob”

Sign

m, sig

Verify

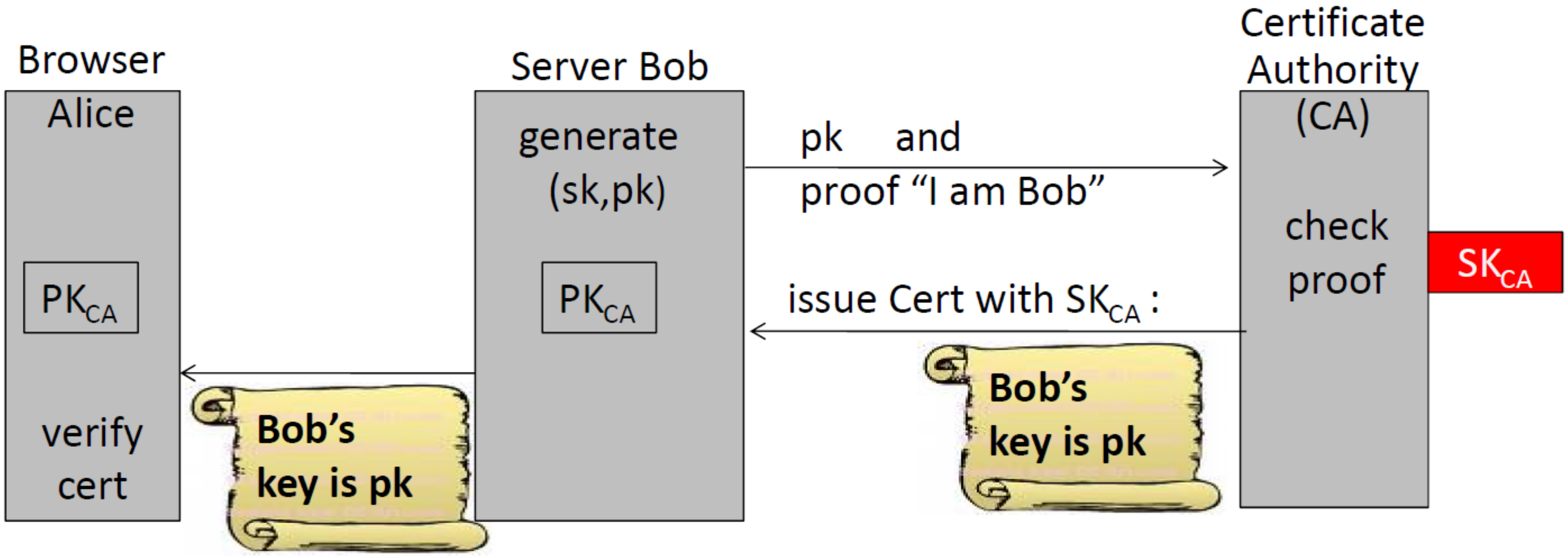
m

Sign(SK_{Alice}, m)=sig

Verify(PK_{Alice}, m)?=sig

PK public key
SK private key

How does Alice obtain Bob's PK?



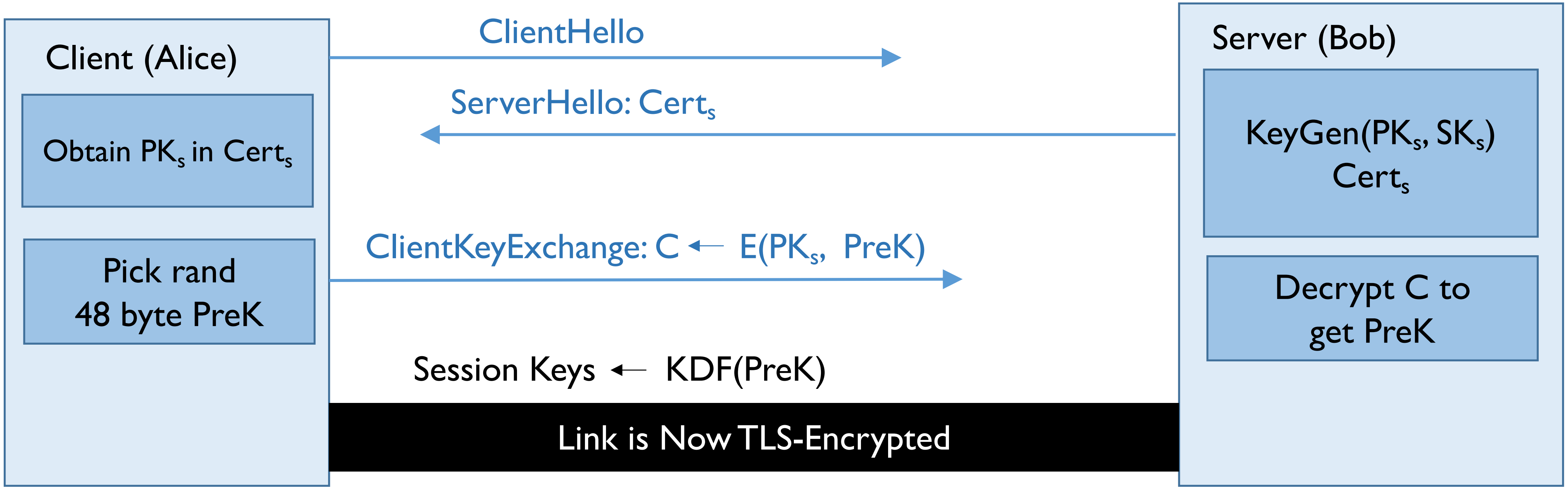
Certificates bind Bob's ID to his PK

Outline

1. Network Security Recap
2. TLS handshake

TLS Handshake v1

- Goal: Establish common session keys



Client (Alice)

Obtain PK_s in $Cert_s$

Pick rand 48 byte PreK

ClientHello

ServerHello: $Cert_s$

ClientKeyExchange: $C \leftarrow E(PK_s, PreK)$

Session Keys ← KDF(PreK)

Link is Now TLS-Encrypted

Server (Bob)

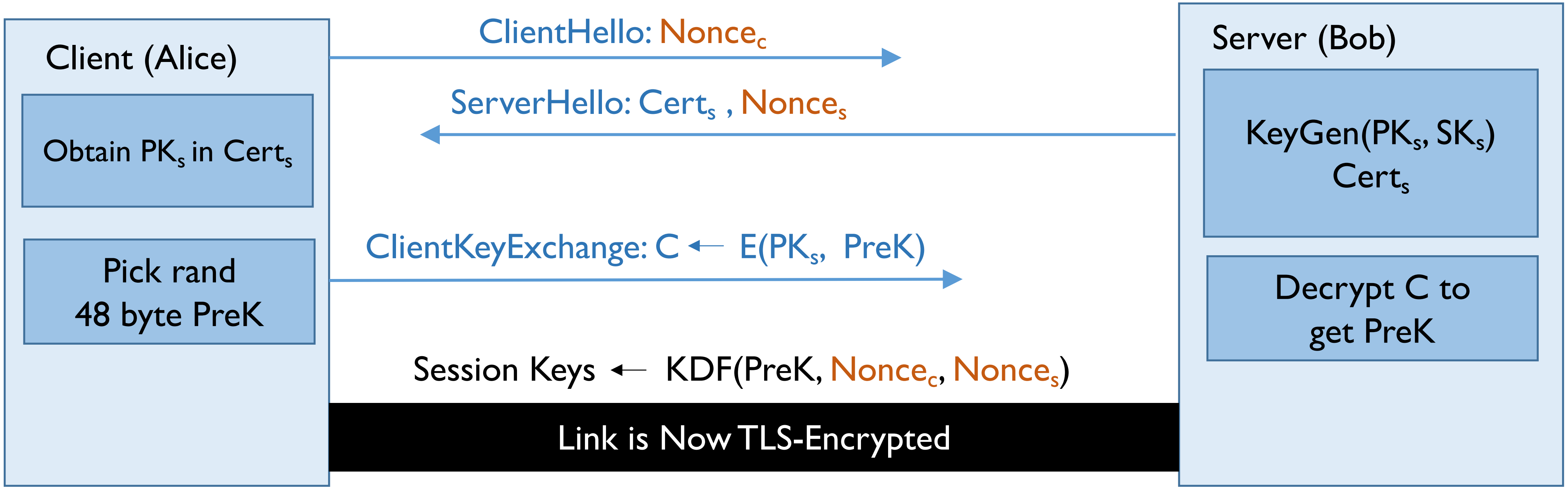
$KeyGen(PK_s, SK_s)$ $Cert_s$

Decrypt C to get PreK

Replay attack can happen!

TLS Handshake v2
Adding randomness protects against replay attack

- Goal: Establish common session keys



Client (Alice)

Obtain PK_s in $Cert_s$

Pick rand 48 byte PreK

ClientHello: $Nonce_c$

ServerHello: $Cert_s$, $Nonce_s$

ClientKeyExchange: $C \leftarrow E(PK_s, PreK)$

Session Keys $\leftarrow$ KDF(PreK, $Nonce_c$, $Nonce_s$)

Link is Now TLS-Encrypted

Server (Bob)

KeyGen(PK_s, SK_s) $Cert_s$

Decrypt C to get PreK

What if SKs gets compromised?

What if Bob’s SK got lost or compromised?

- Bob’s certificate has to be revoked
- Bob regenerates (PK, SK) pair and get a new certificate from CA

If an attacker has recorded past message exchange, he can encrypt with the compromised private key!

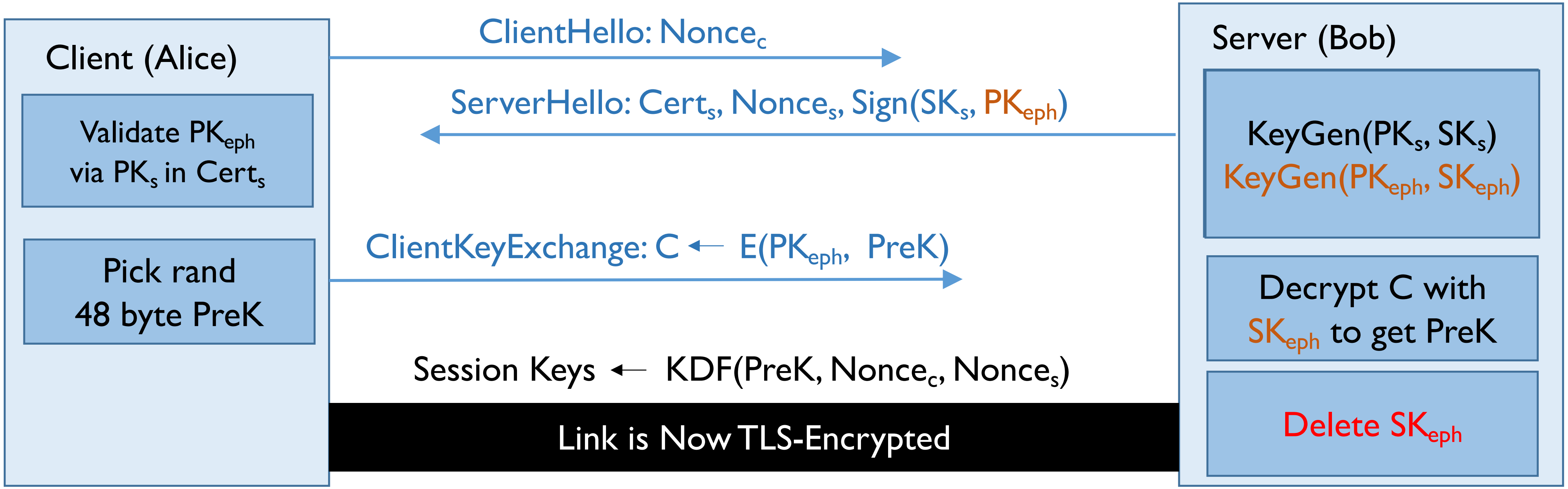
Key exchange should provide forward secrecy

Future compromise of secret key should NOT affect past sessions

- Need a separate session key other than the private key
- Computationally less burdensome

TLS Handshake with forward secrecy

- Goal: Establish common session keys



Client (Alice)

Validate PK_{eph} via PK_s in $Cert_s$

Pick rand 48 byte PreK

ClientHello: $Nonce_c$

ServerHello: $Cert_s$, $Nonce_s$, Sign(SK_s, PK_{eph})

ClientKeyExchange: C ← E(PK_{eph}, PreK)

Session Keys ← KDF(PreK, $Nonce_c$, $Nonce_s$)

Link is Now TLS-Encrypted

Server (Bob)

KeyGen(PK_s, SK_s)
KeyGen(PK_{eph}, SK_{eph})

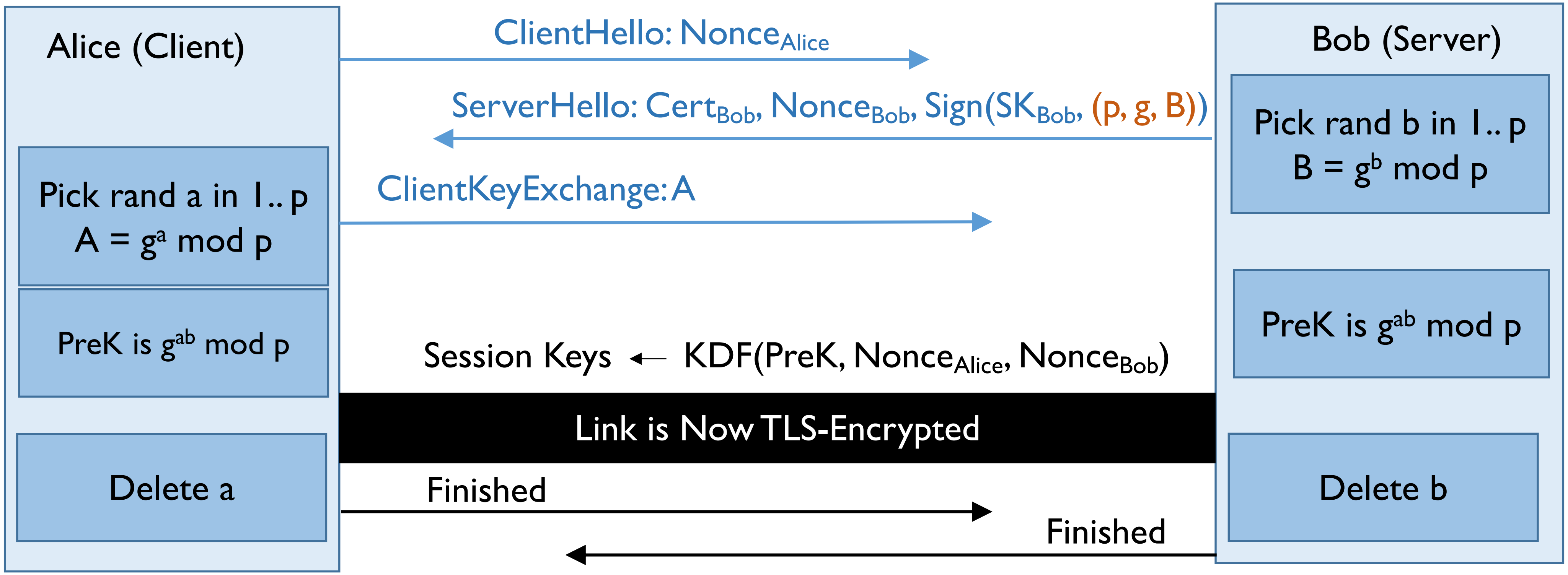
Decrypt C with SK_{eph} to get PreK

Delete SK_{eph}

RSA Key Gen is Slow. Can we do better?

TLS Handshake via Diffie Hellman

- Goal: Symmetric key exchange



Alice (Client)

Pick rand a in 1.. p
$A = g^a \bmod p$

PreK is $g^{ab} \bmod p$

Delete a

ClientHello: $\text{Nonce}_{\text{Alice}}$

ServerHello: Cert_{Bob}, $\text{Nonce}_{\text{Bob}}$, Sign($\text{SK}_{\text{Bob}}$, (p, g, B))

ClientKeyExchange: A

Session Keys ← KDF(PreK, $\text{Nonce}_{\text{Alice}}$, $\text{Nonce}_{\text{Bob}}$)

Link is Now TLS-Encrypted

Finished

Finished

Bob (Server)

Pick rand b in 1.. p
$B = g^b \bmod p$

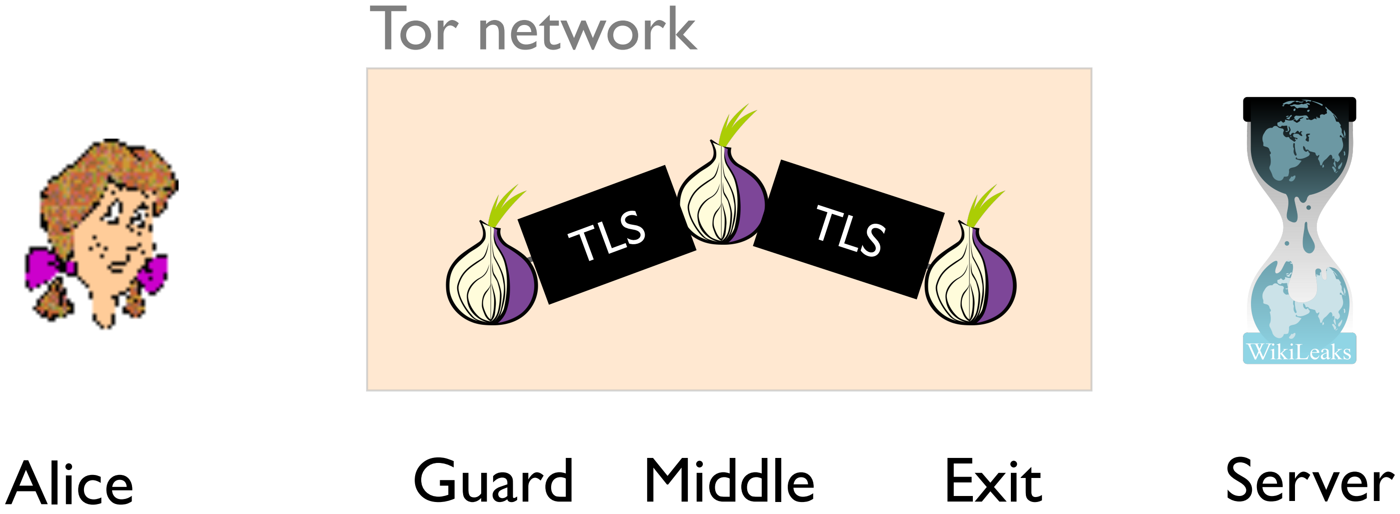
PreK is $g^{ab} \bmod p$

Delete b

Outline

1. Network Security Recap
2. TLS handshake
3. The full story of Tor Circuit

TLS connections are pre-established among Tor nodes



Tor network

TLS

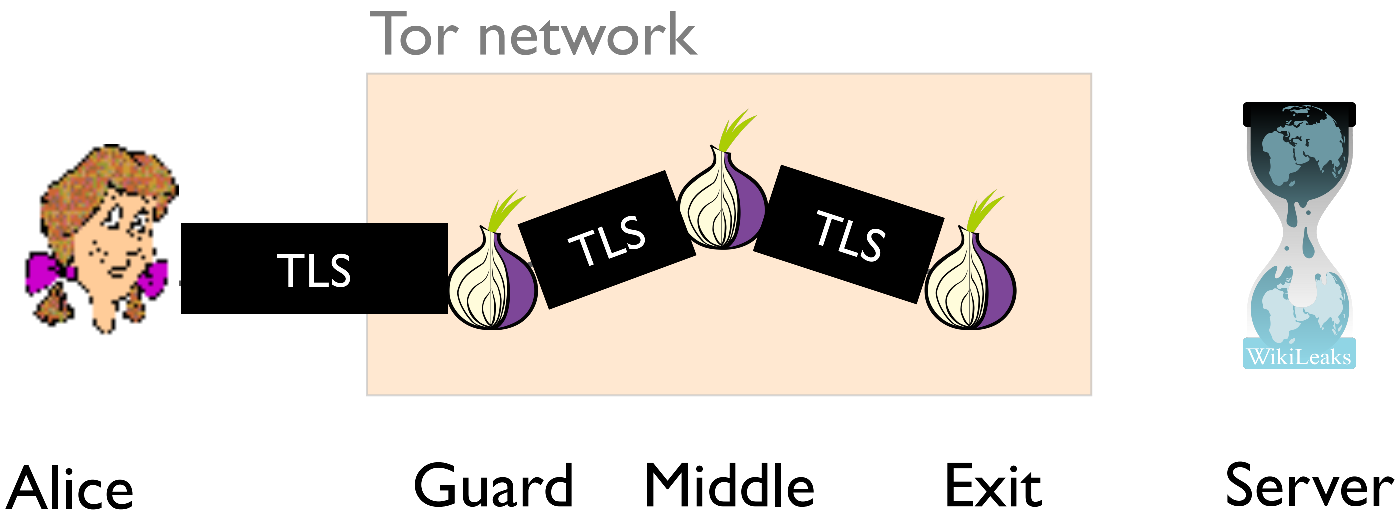
TLS

Alice

Guard Middle Exit

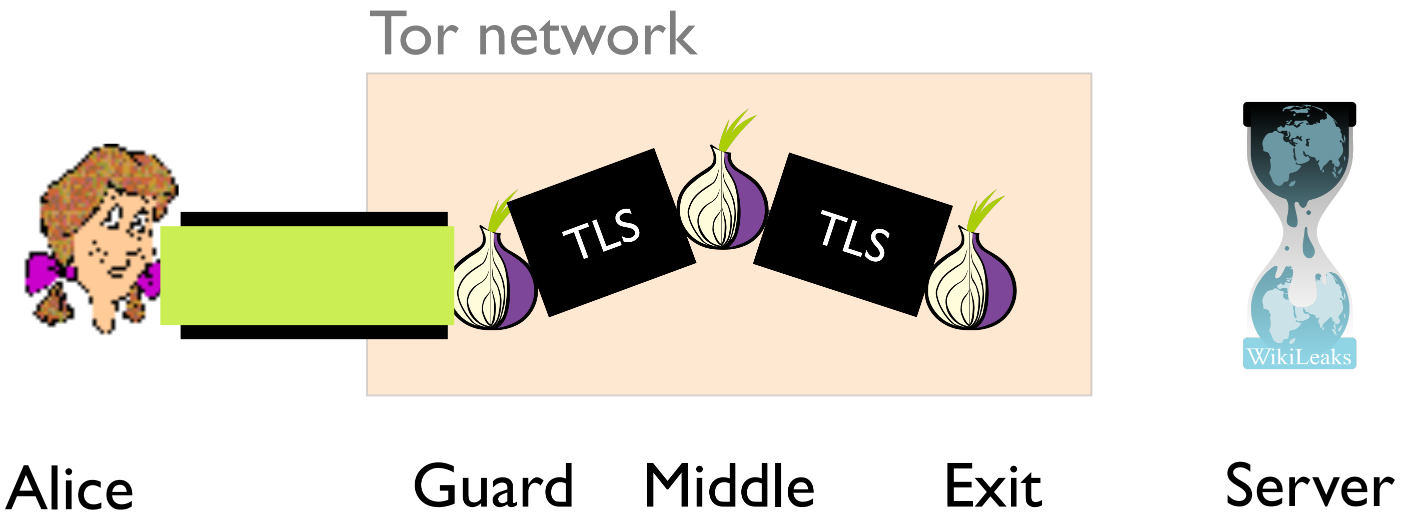
Server

TLS connection first needs to be established between Alice and Guard



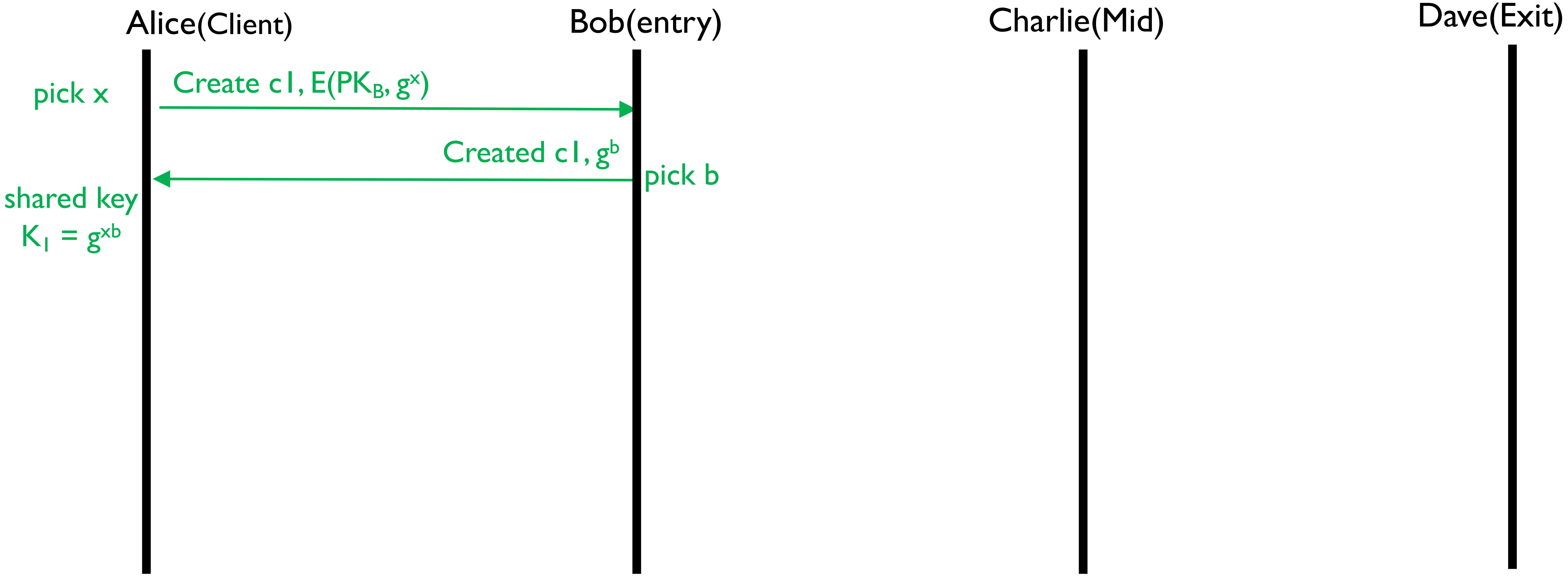
ANY messages exchanged between each connection
is encrypted using the set of session keys (connection key in Tor)

With TLS tunnel already established
Alice starts the steps to build the Tor circuit

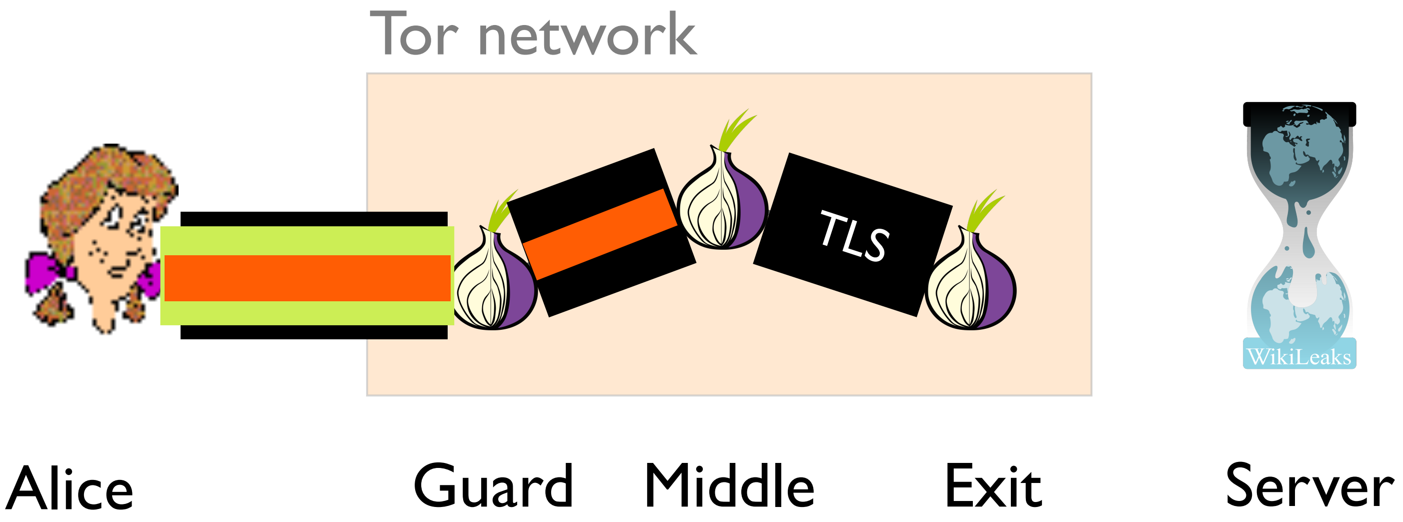


Tor Circuit Construction: 1st hop

- How Alice – Bob establish shared session key K_1

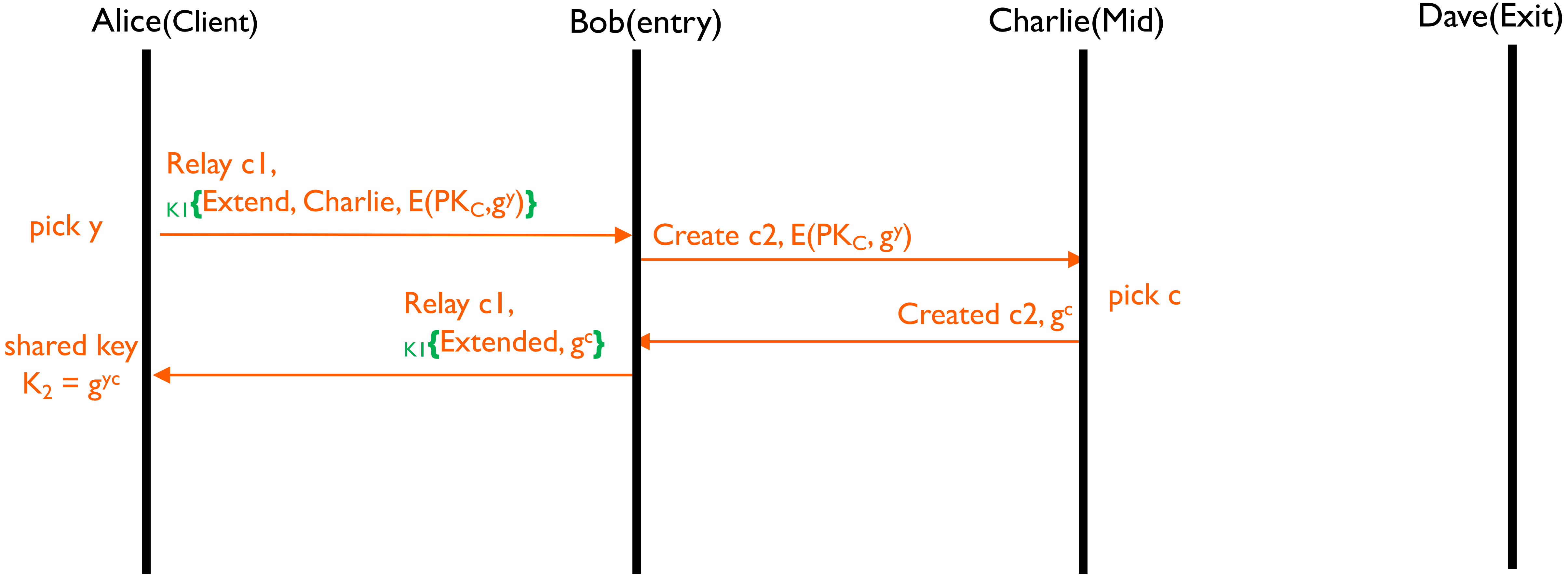


With TLS tunnel already established
Alice starts the steps to build the Tor circuit



Tor Circuit Construction: 2nd hop

- How Alice – Charlie establish shared session key K_2



Alice(Client)

Bob(entry)

Charlie(Mid)

Dave(Exit)

pick y

Relay c1, ${}_{K1}\{$Extend, Charlie, E(PK_C,g^y)$\}$

Create c2, E(PK_C, g^y)

pick c

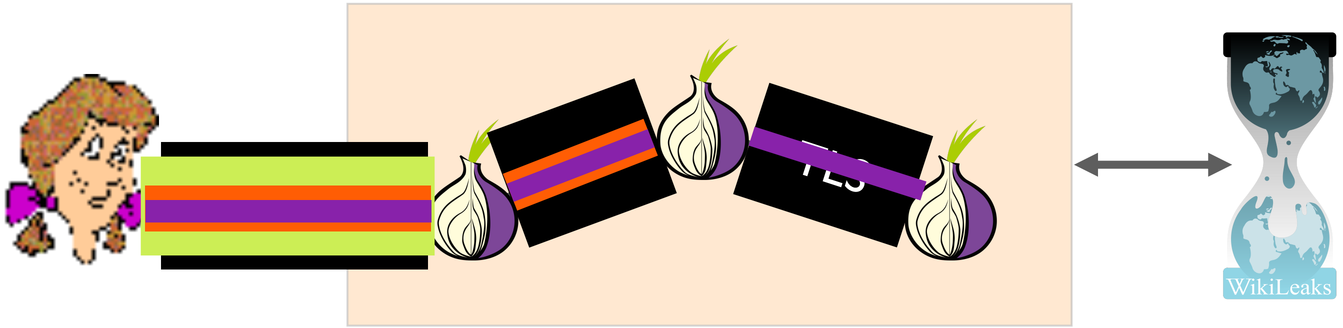
Created c2, g^c

Relay c1, ${}_{K1}\{$Extended, $g^c$$\}$

shared key
$K_2 = g^{yc}$

With TLS tunnel already established
Alice starts the steps to build the Tor circuit

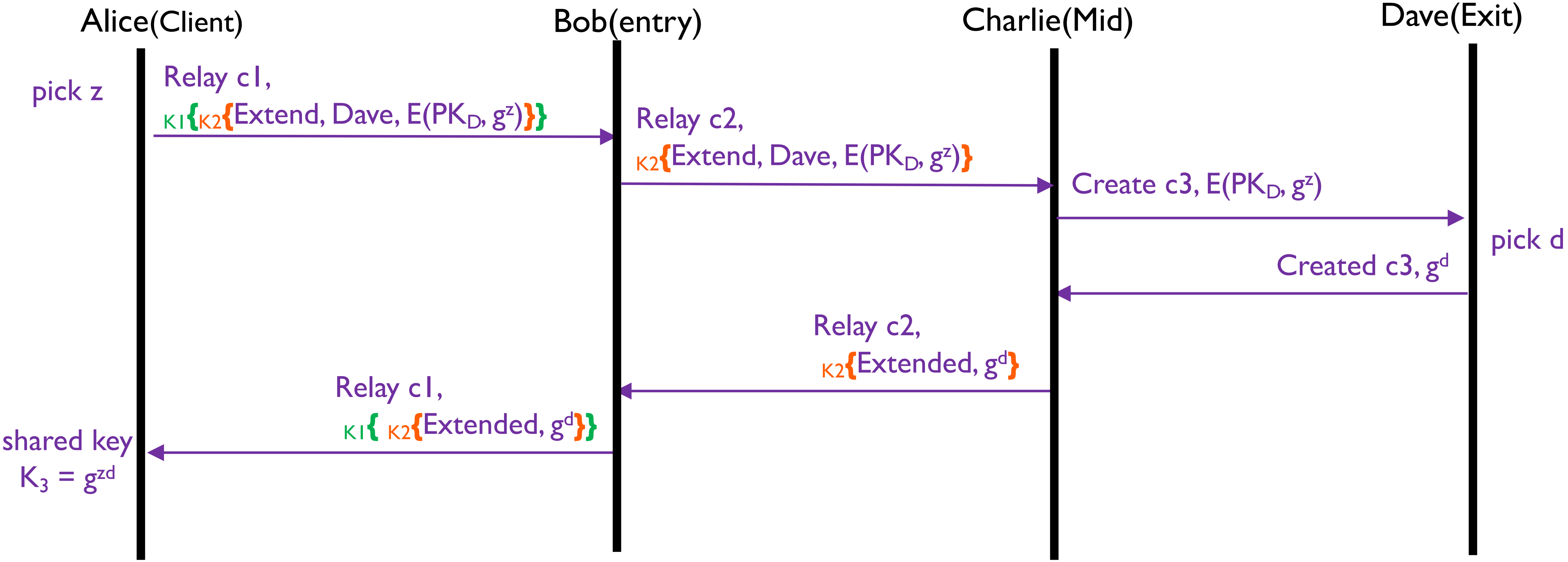
Tor network



Alice Guard Middle Exit Server

Tor Circuit Construction: 3rd hop

- How Alice – Dave establish shared session key K_3



Alice(Client)

Bob(entry)

Charlie(Mid)

Dave(Exit)

pick z

Relay c1, $_{K1}\{_{K2}\{\text{Extend, Dave, } E(PK_D, g^z)\}\}$

Relay c2, $_{K2}\{\text{Extend, Dave, } E(PK_D, g^z)\}$

Create c3, $E(PK_D, g^z)$

pick d

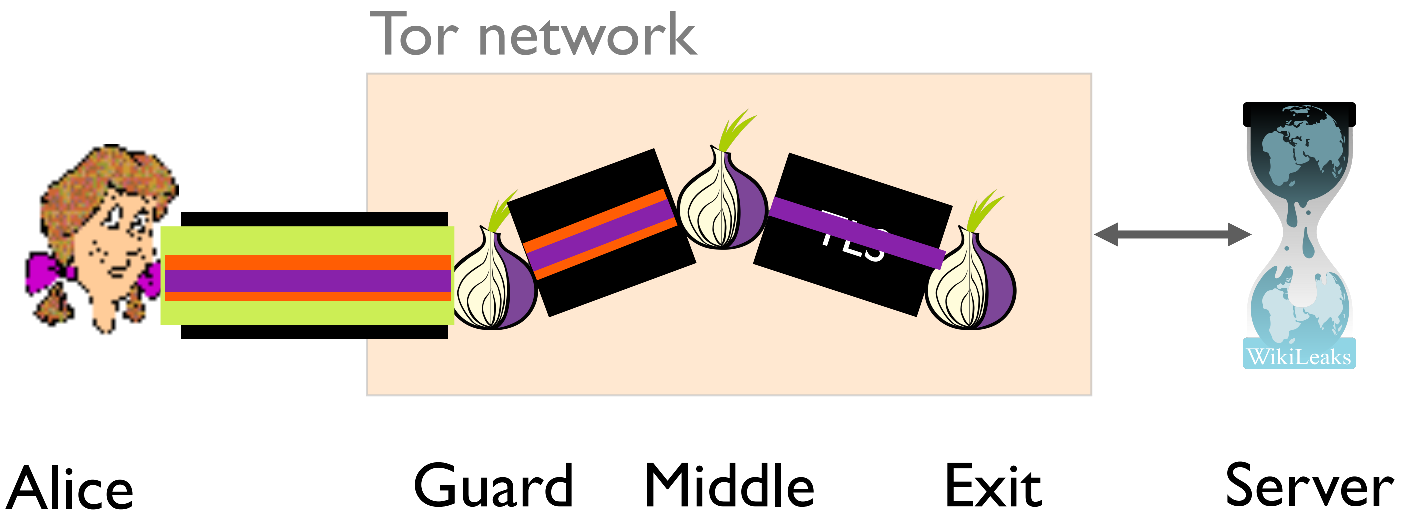
Created c3, g^d

Relay c2, $_{K2}\{\text{Extended, } g^d\}$

Relay c1, $_{K1}\{\ _{K2}\{\text{Extended, } g^d\}\}$

shared key $K_3 = g^{zd}$

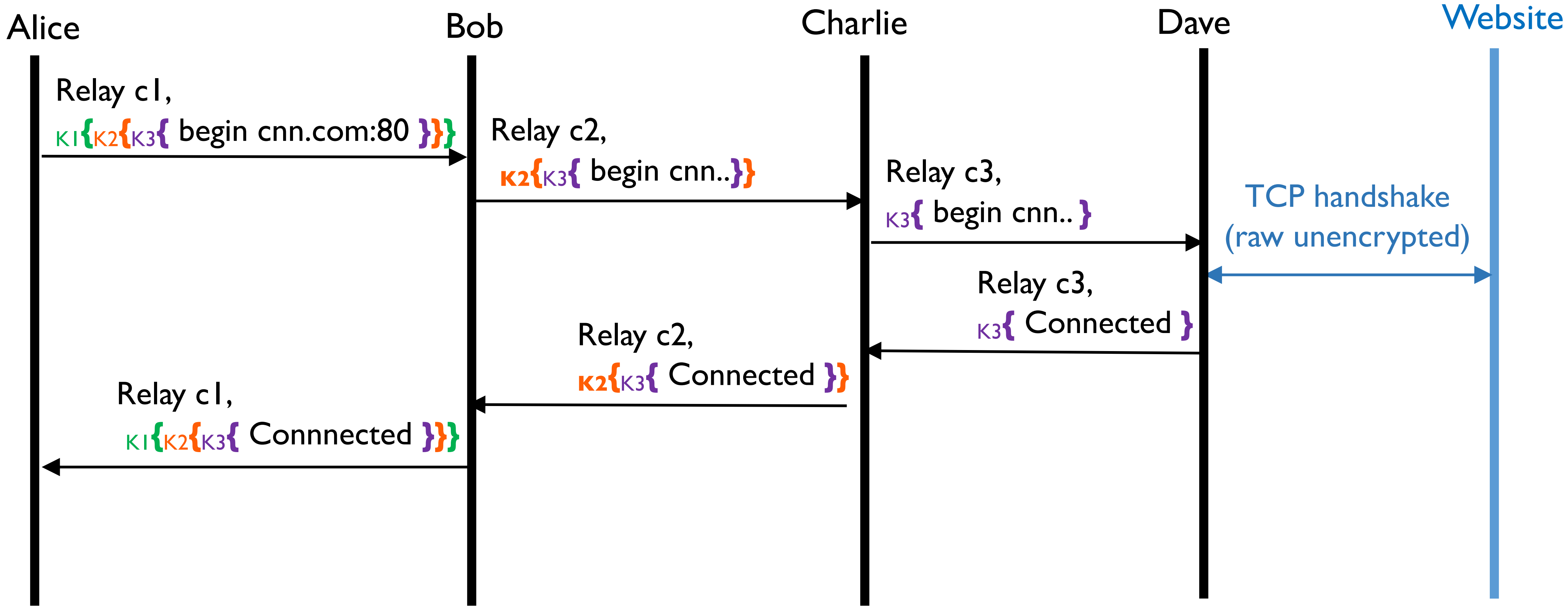
ALL Tor messages are exchanged inside TLS tunnels



This makes it hard to distinguish Tor traffic from normal TLS traffic

Tor Packet Forwarding via 3 hop Circuit

- Alice – Bob, Alice – Charlie, Alice – Dave has shared session key K_1, K_2 and K_3



When selecting relays what should Alice consider?

Alice (Client) → Bob (Entry) → Charlie (Middle) → Dave (Exit) → Server

Diversify the relays as much as possible! Why?

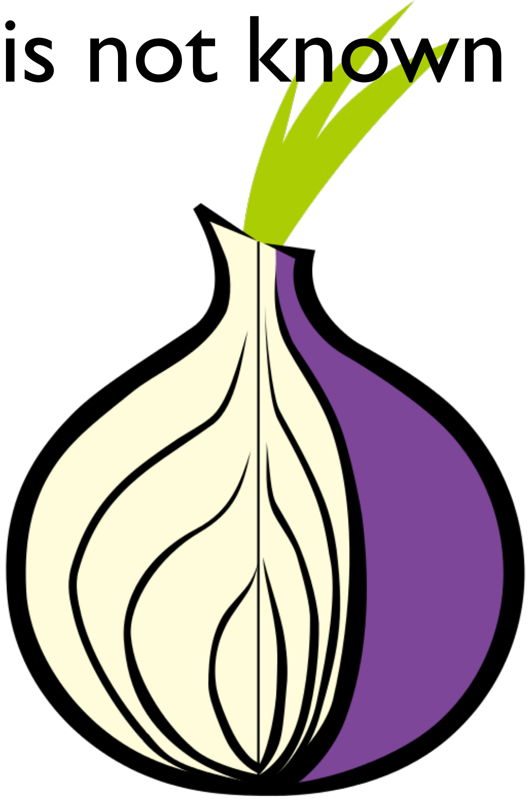
Top Countries where Tor relays are located

- The US
- Germany
- France
- Russia
- Netherlands
- United Kingdom

Outline

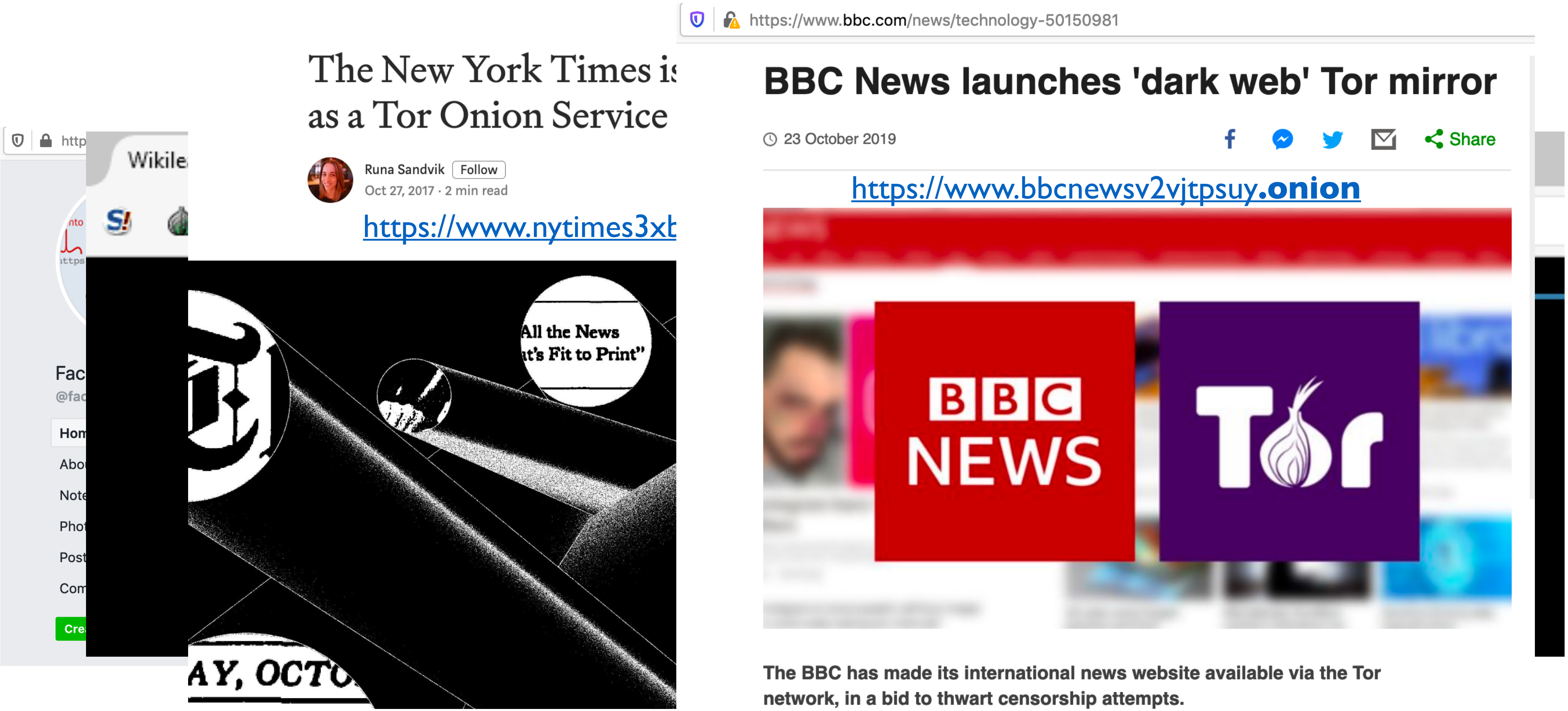
1. Network Security Recap
2. TLS handshake
3. The full story of Tor Circuit
4. 🤘 Tor Onion Service (aka hidden service)

Motivation: Now that we have secured Alice (identity, IP, location) is not known to server



Can we hide the IP and location of the server from Alice?

Tor Onion Service (aka Hidden Service)



The New York Times is
as a Tor Onion Service

Runa Sandvik Follow
Oct 27, 2017 · 2 min read

https://www.nytimes3xb

https://www.bbc.com/news/technology-50150981

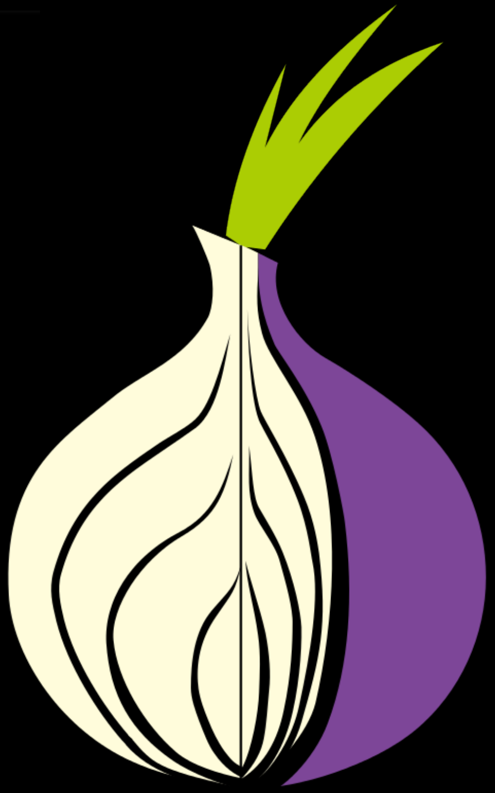
BBC News launches 'dark web' Tor mirror

23 October 2019 Share

https://www.bbcnewsv2vjtpsuy.**onion**

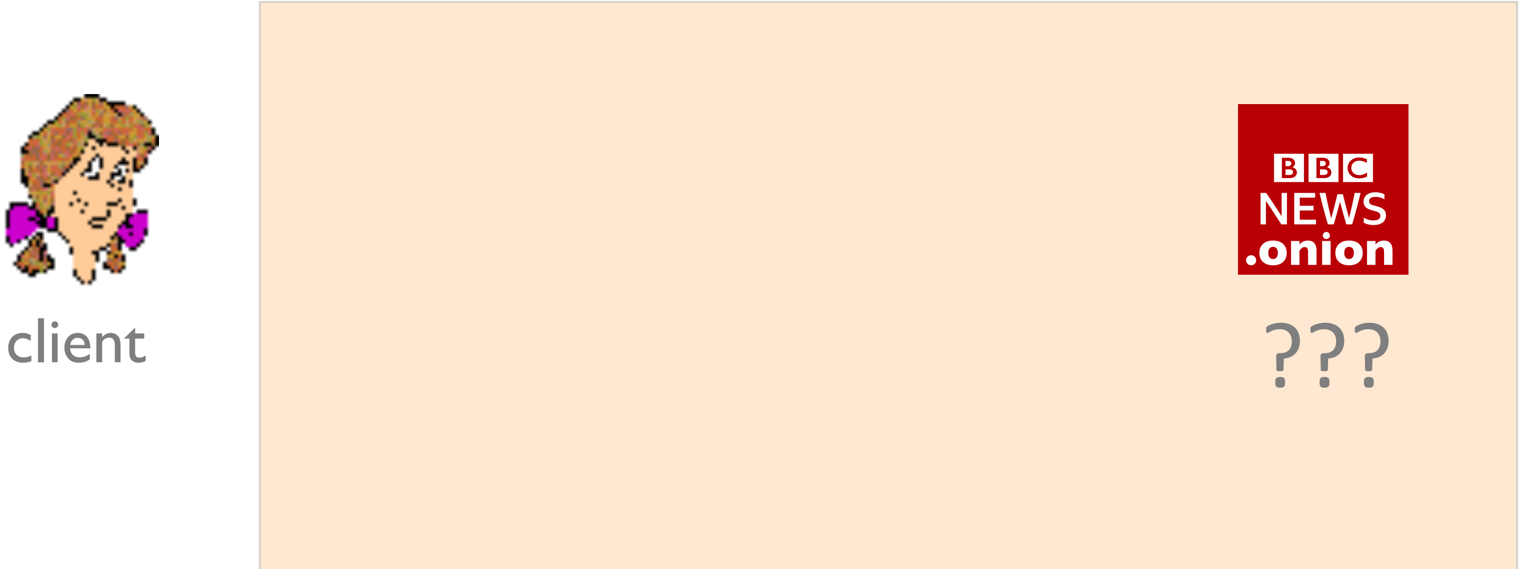
The BBC has made its international news website available via the Tor network, in a bid to thwart censorship attempts.

WE ARE
ANONYMOUS



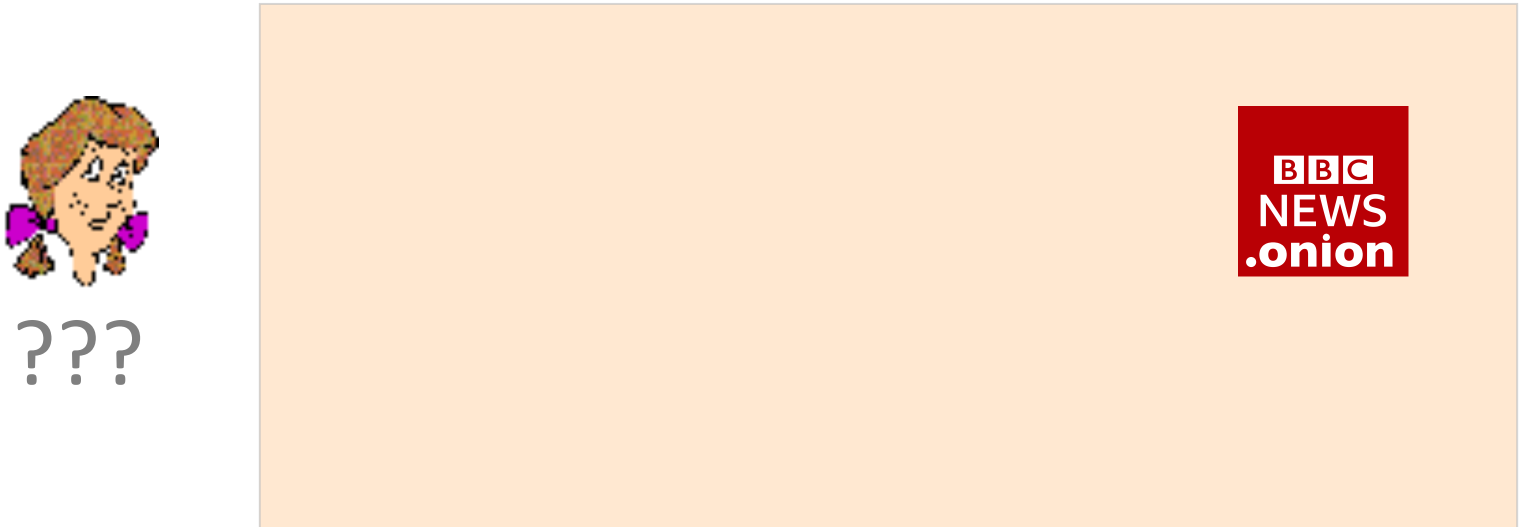
Onion service provides server anonymity by concealing server IP and location

Alice (client) shouldn't know where onion service (server) is



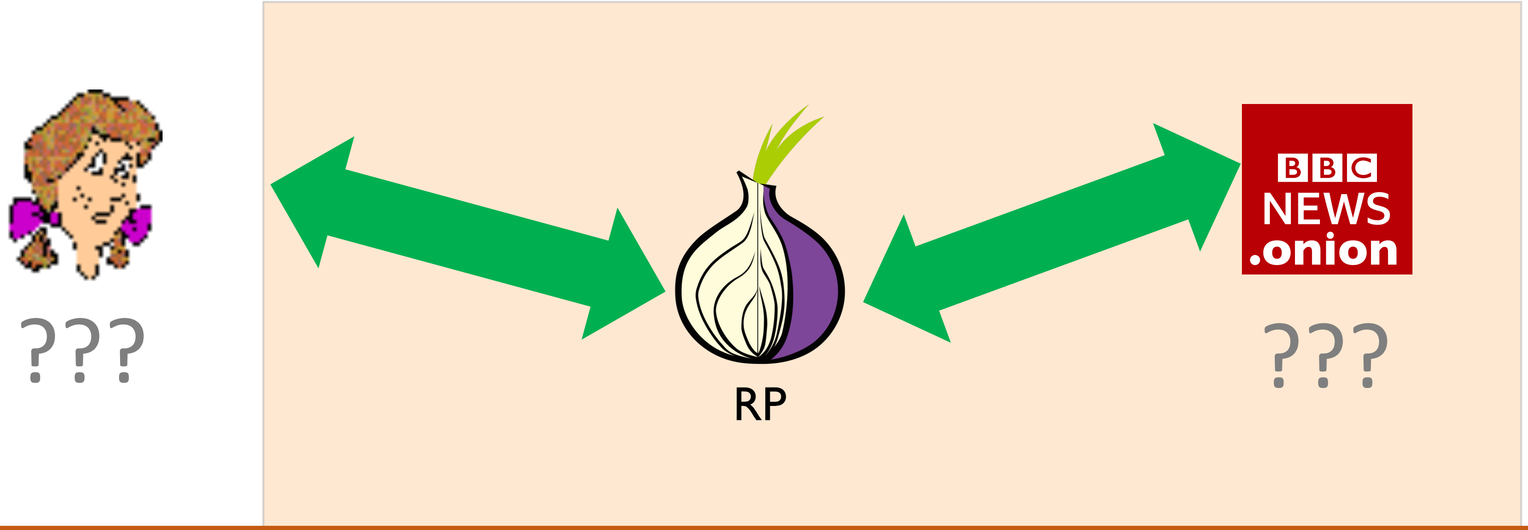
Useful for servers hosting sensitive information

From server's point of view
Alice should also remain anonymous



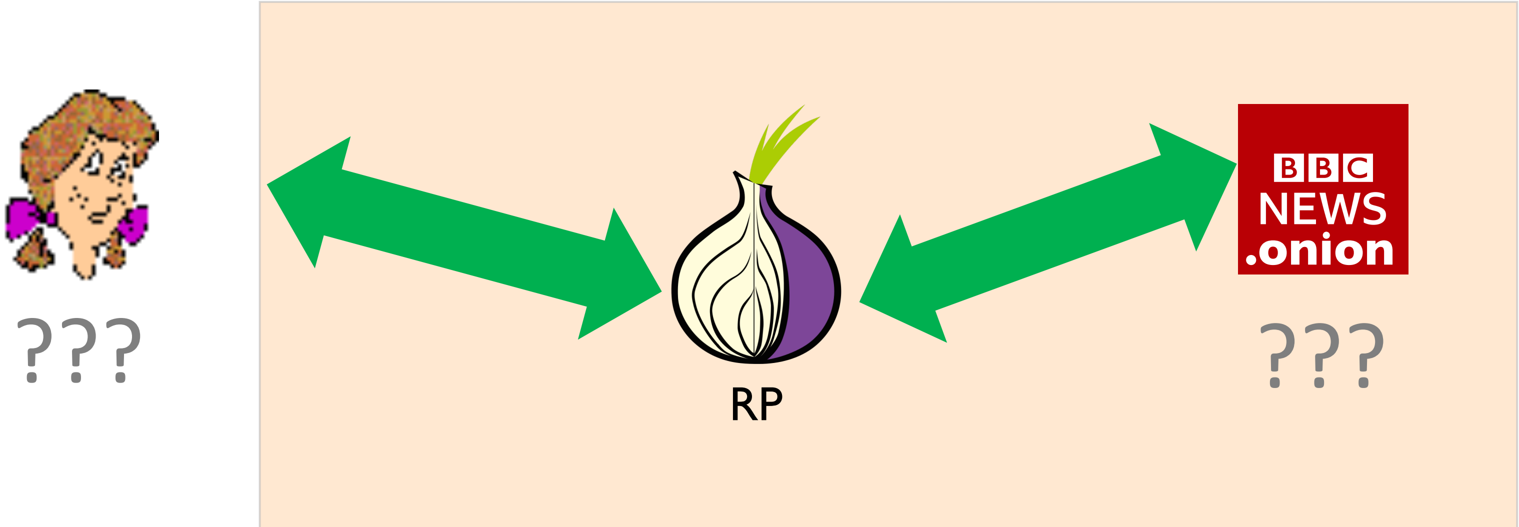
How to achieve both client-side and server-side anonymity?

A middleman between Alice and onion service is needed: Tor calls it a Rendezvous Point (RP)



They DON'T! RP should never learn anything regarding both Alice or server

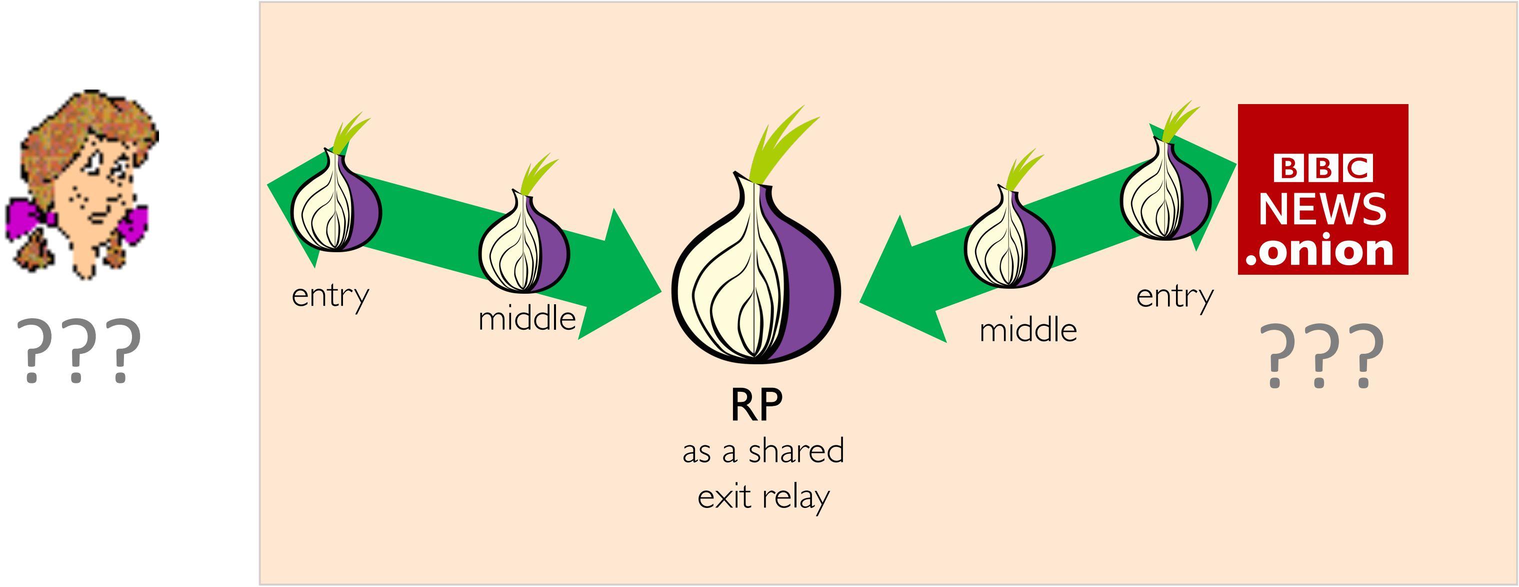
How many hops should RP have from Alice and Server?



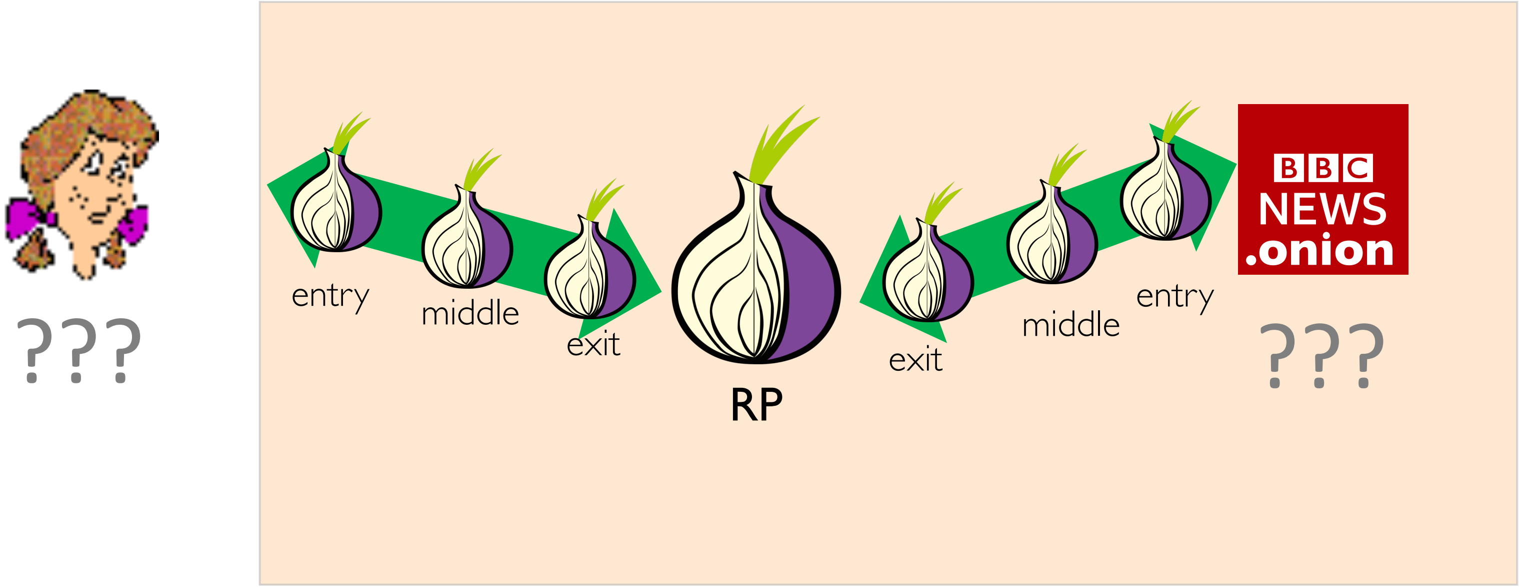
3 hop is required for anonymity for both Alice and Server

How about 5 hop topology?

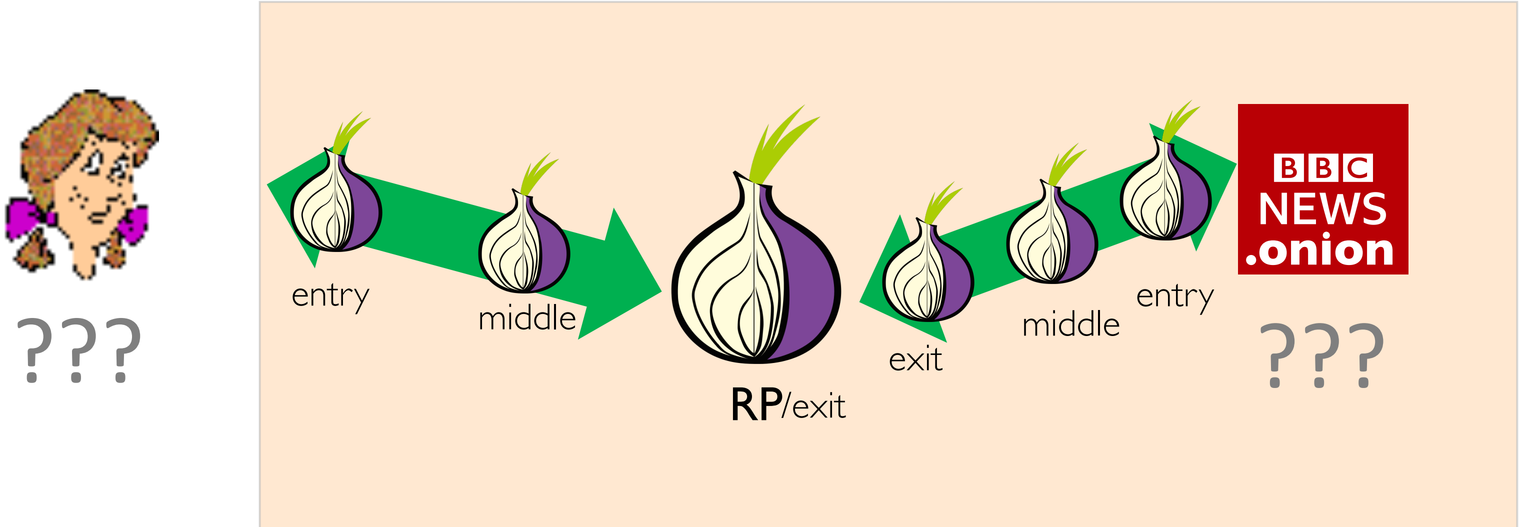
RP is exactly 3 hop away from both Alice and Server



How about 7 hop topology?

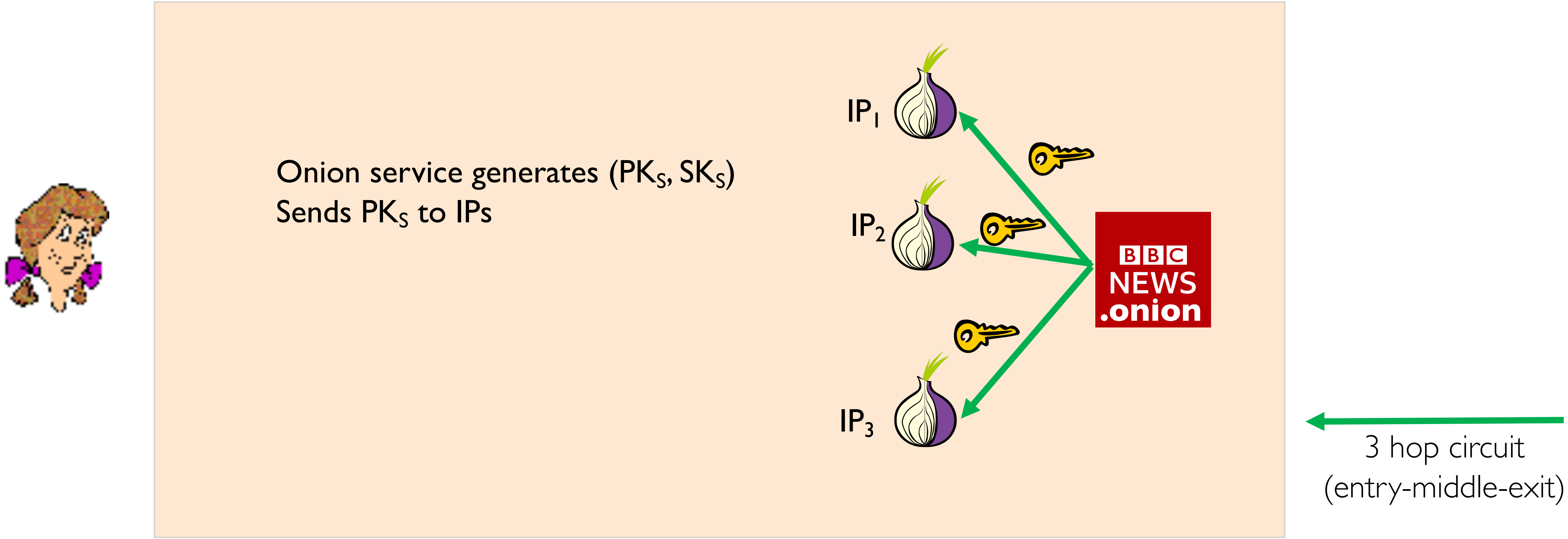


RP should be at least 3 hops away from both client and server without any overlap to support bi-directional anonymity



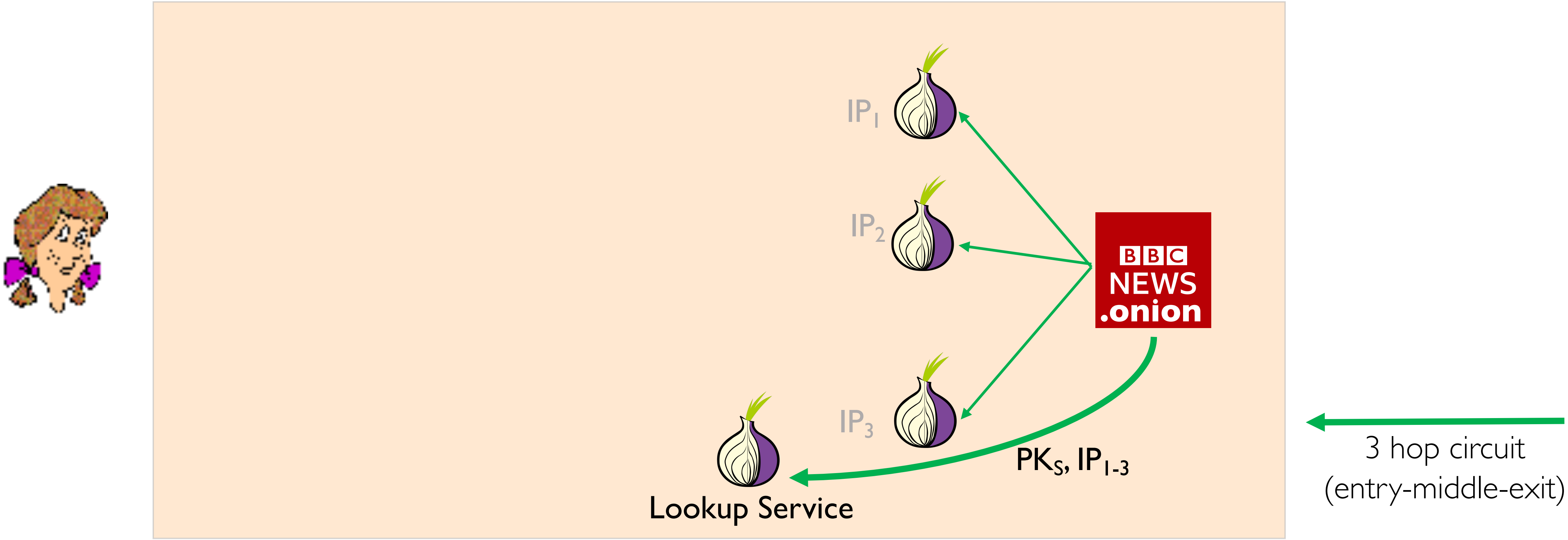
How to agree on RP without exposing oneself?

Step 1: Server picks random 3 relays as its introduction points(IP) and builds circuits to them

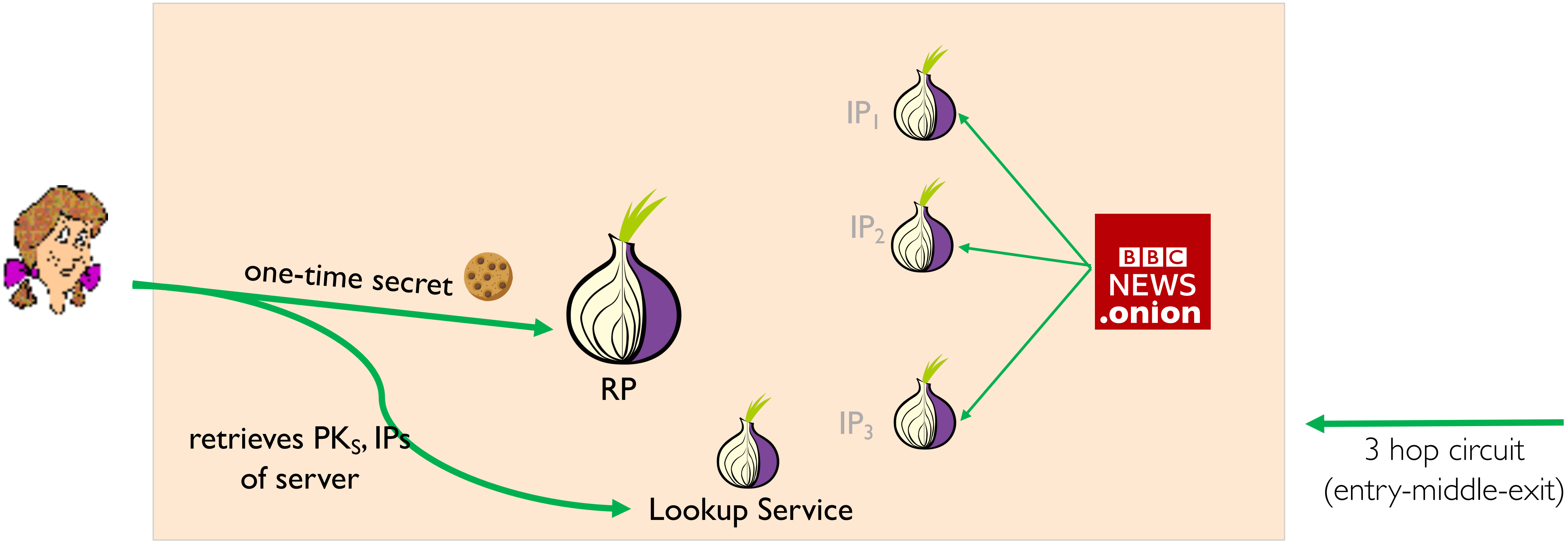


Onion service generates (PK_S, SK_S)
Sends PK_S to IPs

Step 2: Server advertises its onion address, PK, and IPs to lookup service

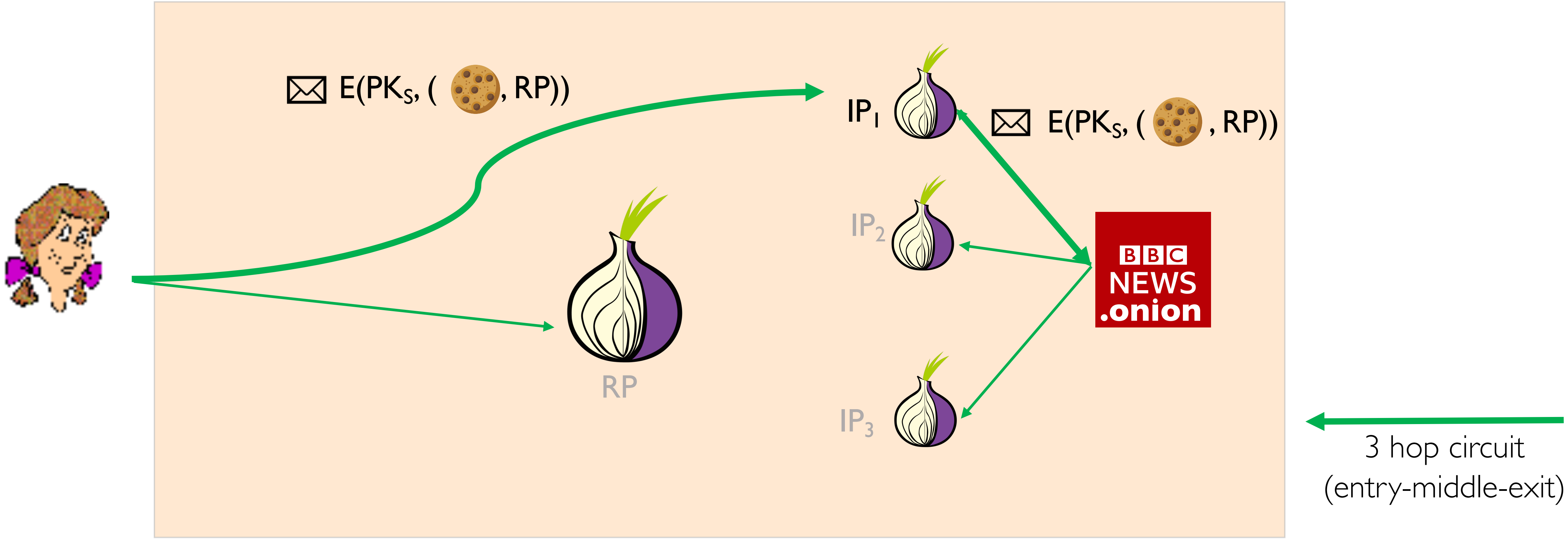


Step 3: Client retrieves the PK, and IPs for the server Also client builds circuit to a randomly chosen RP

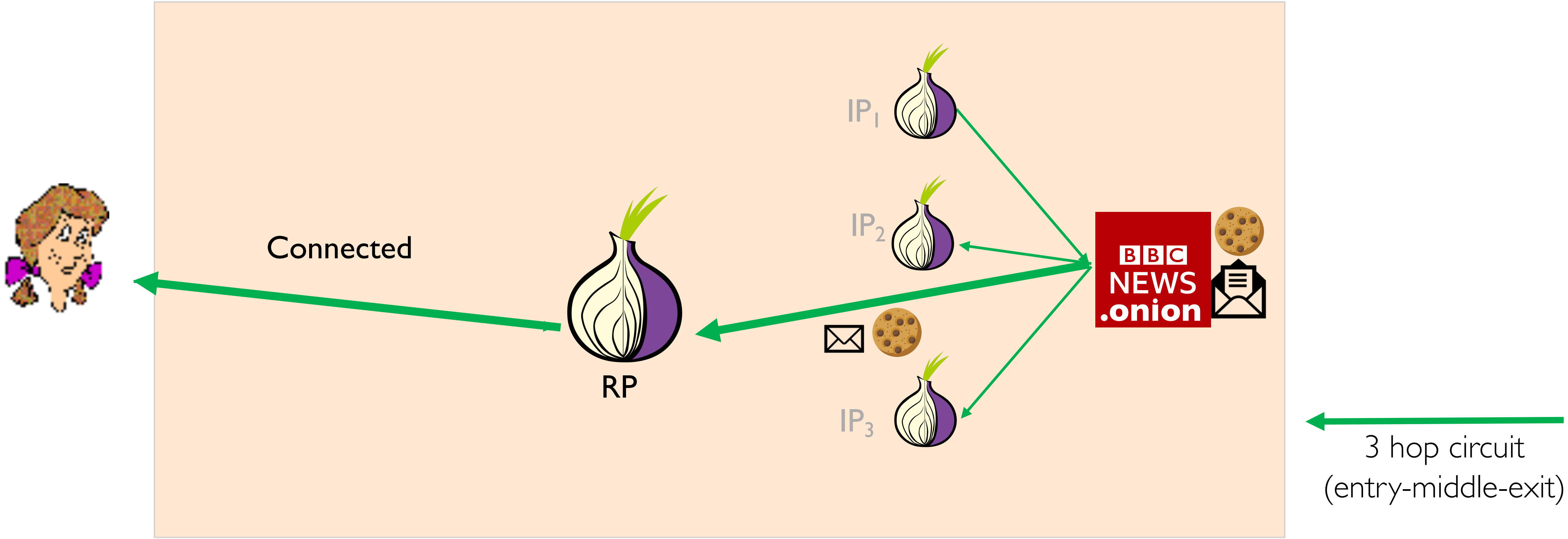


one-time secret
RP
retrieves PK_S, IPs of server
Lookup Service
IP_1
IP_2
IP_3
BBC NEWS .onion
3 hop circuit (entry-middle-exit)

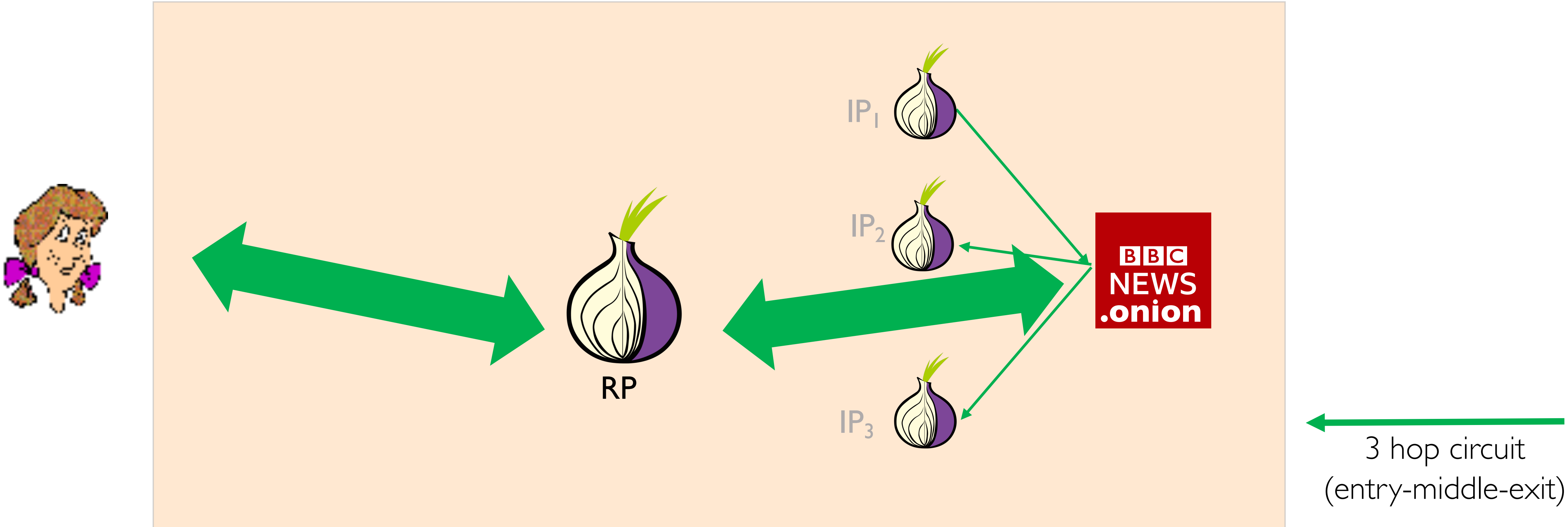
Step 4: Client sends *introduce message* to server via IP



Step 5: Server sends *rendezvous message* to RP

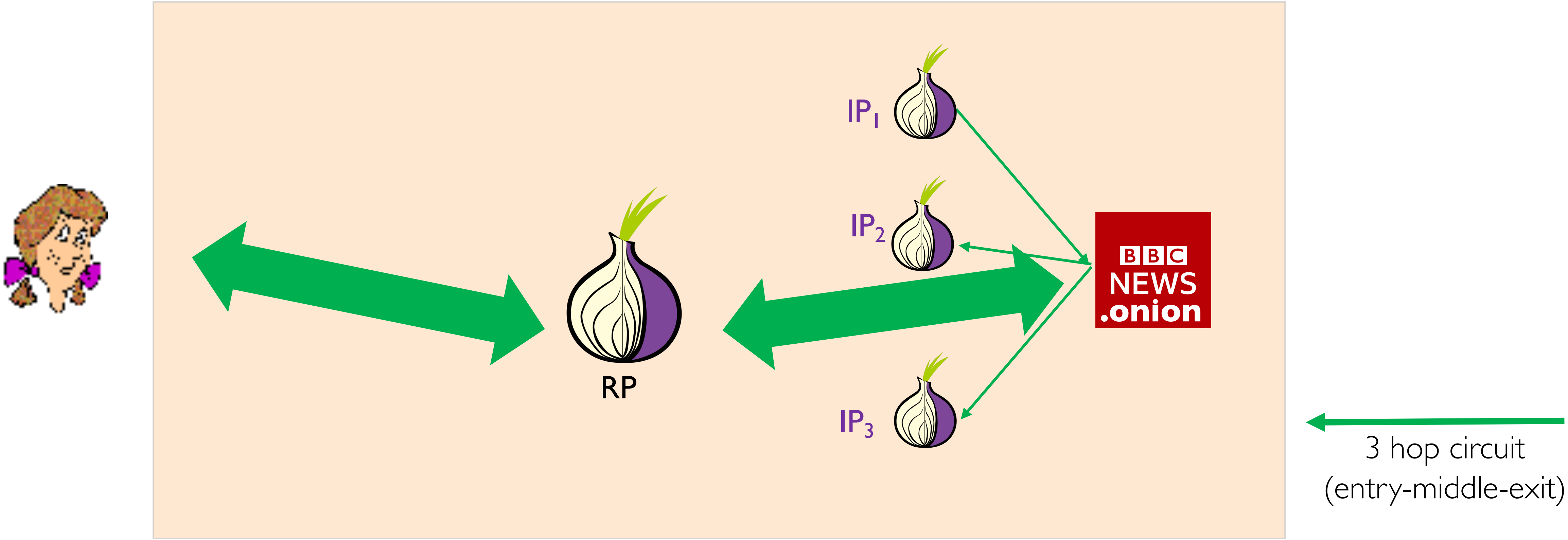


Step 6: Client and server proceeds to use Tor circuits like normal



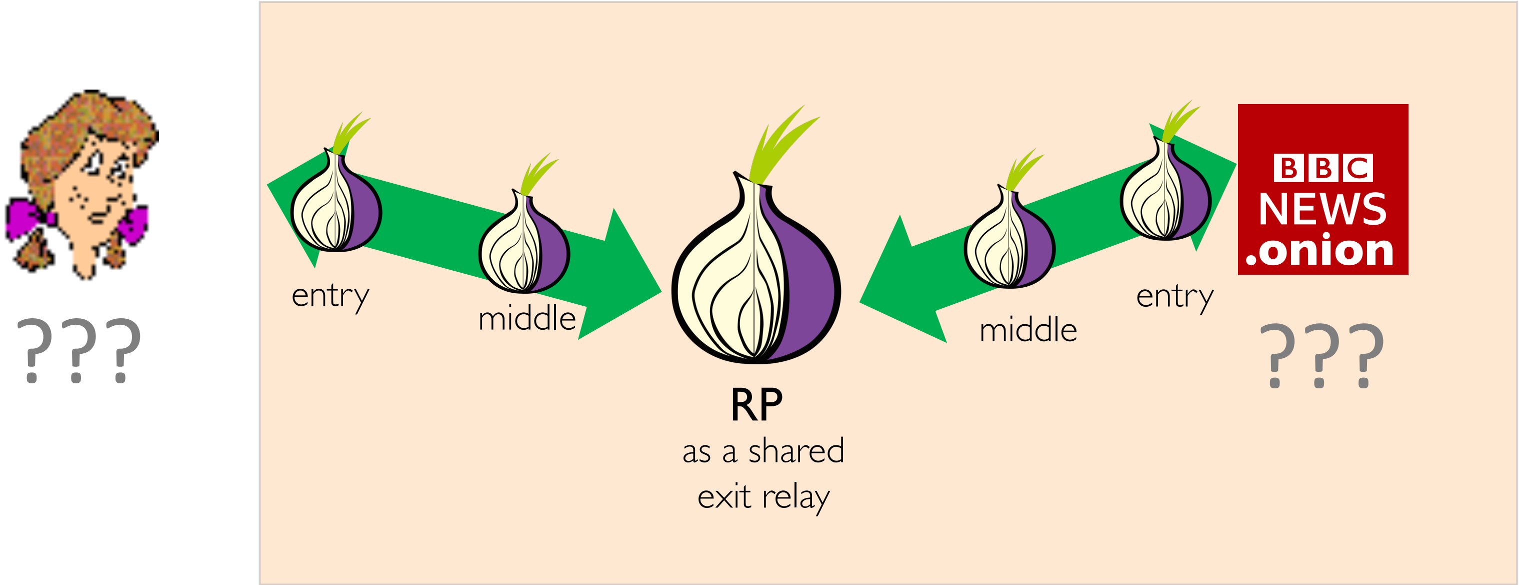
None of IPs, RP, and LS do not know about server or client IP/location

Why can’t just IPs be the RP forwarding data for server?



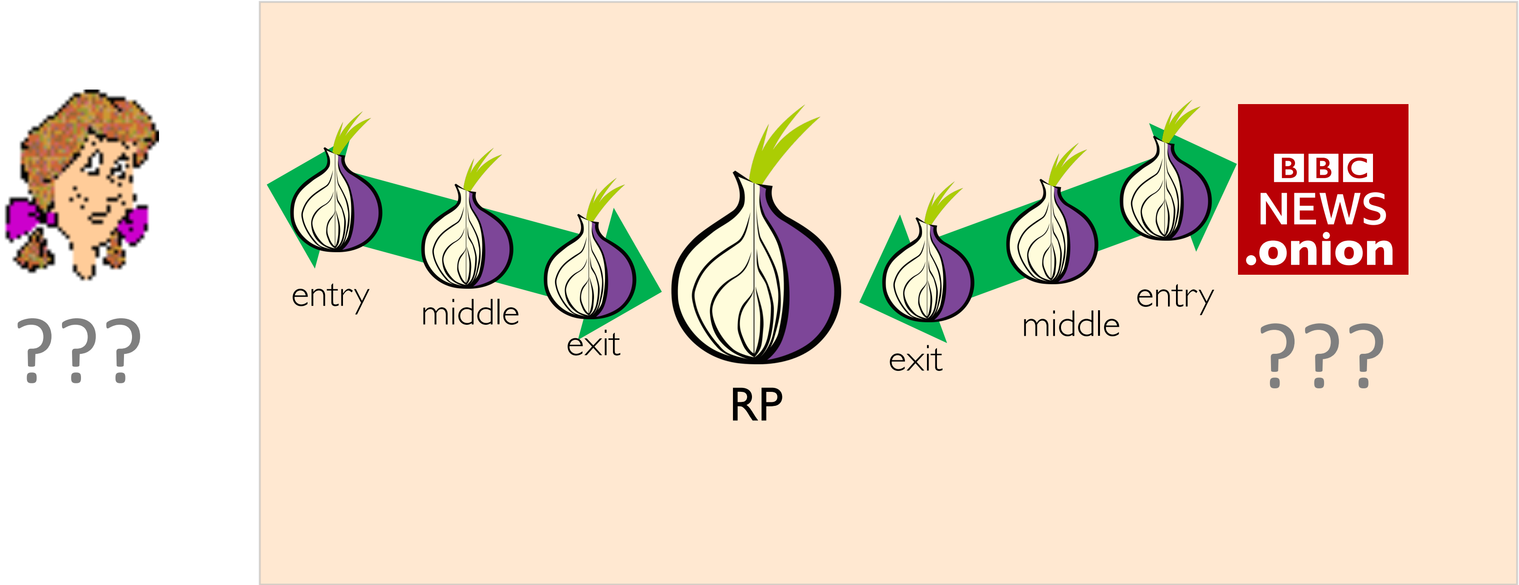
How about 5 hop topology?

If RP is compromised, then both circuits are impacted

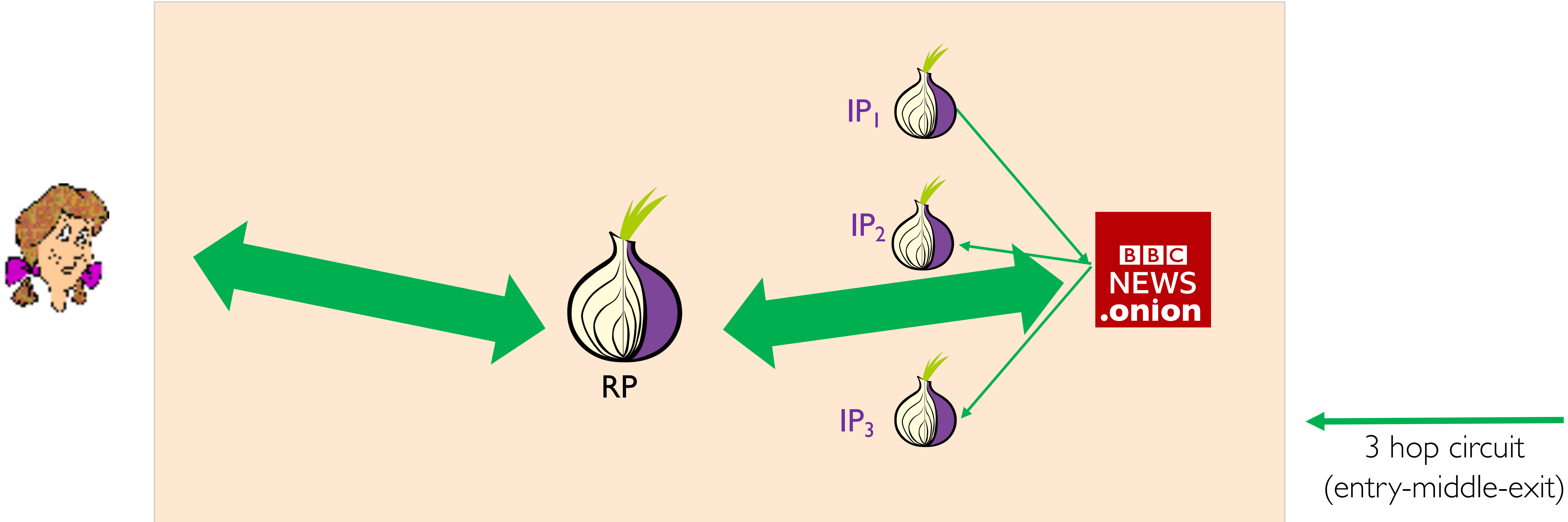


7 hop works but unnecessary as RP is simply forwarding

No added value in terms of security but only causes longer delay



Why can’t just IP be the RP forwarding data for server?



Having a separate RP per client helps spreading the load over different RPs

WE ARE NOT SO
ANONYMOUS

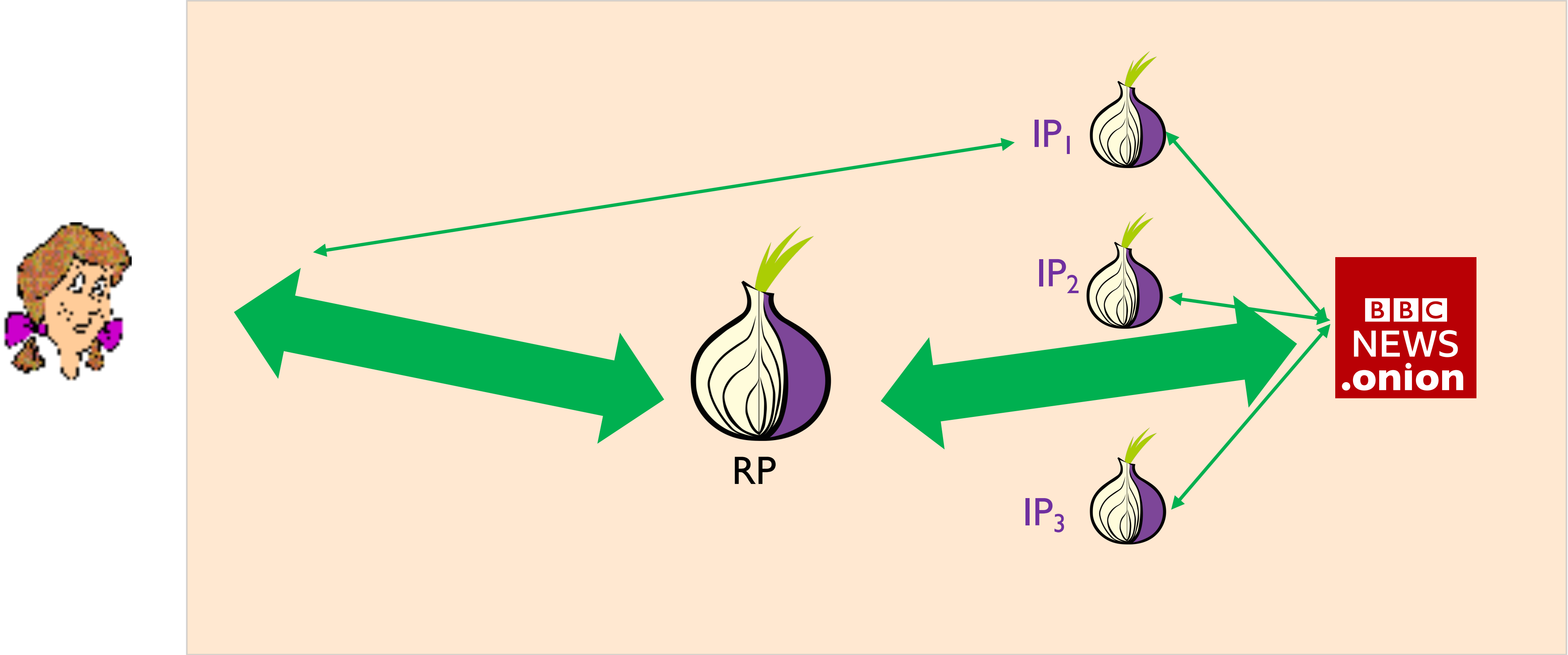
Outline

1. Network Security Recap
2. TLS handshake
3. The full story of Tor Circuit
4. Tor Onion Service (aka hidden service)
5. When Tor hidden service is not really hidden

Fingerprinting Attacks

Circuit Fingerprinting Attack:
Passive Deanonymization of Tor Hidden Service (USENIX Sec'15)

Circuits for onion service has unique characteristics



Circuits for onion service has unique characteristics

- HS-IP circuits are long-lived while Client-IP circuits are short-lived
- IP’s have little incoming and outgoing cells
- HS-RP circuits have more outgoing than incoming
- Streams for different .onion domains are not multiplexed
- IP and RP circuits are disjoint from general circuits

Use these characteristics to identify
onion service circuits and locate the server!

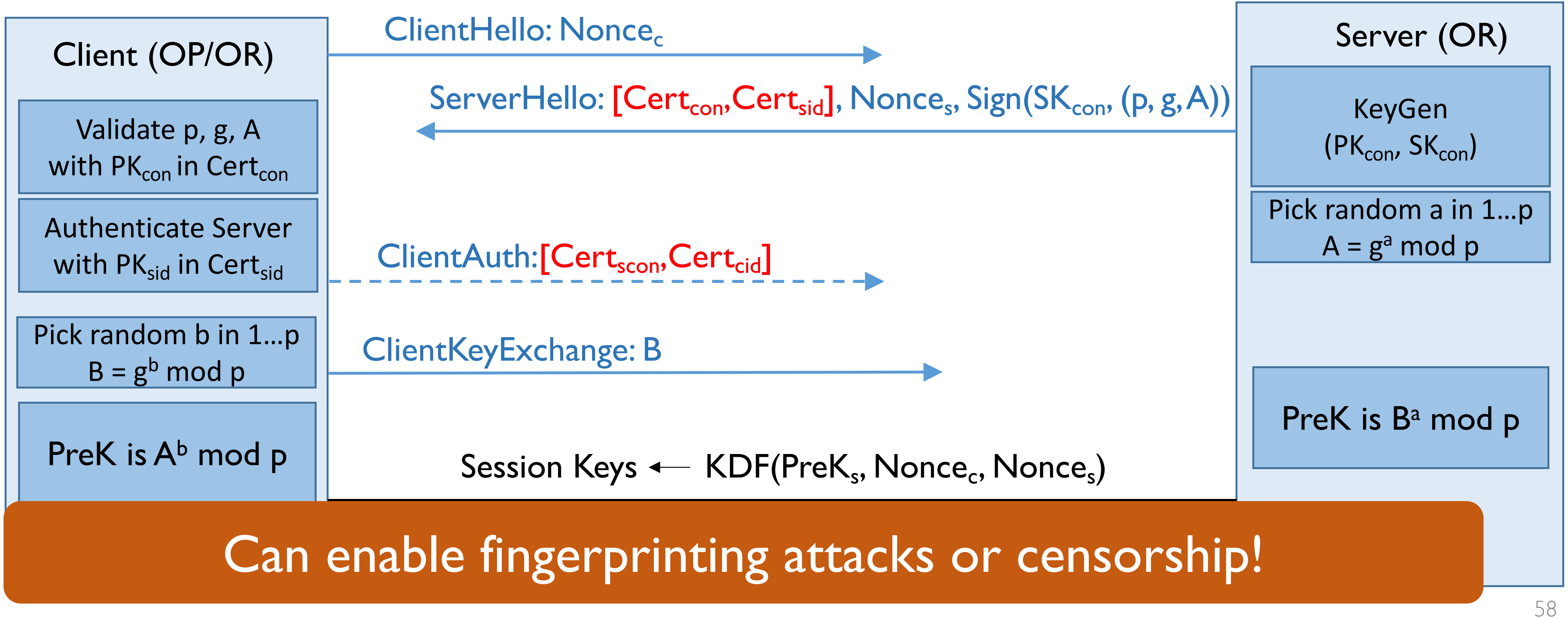
Summary of Tor

- Tor enables anonymous communication over the Internet
- Tor uses 3 hop encrypted circuit to provide anonymity
- Tor Onion service aims to achieve both client-server server-client anonymity by hiding server IP/location
- Tor is vulnerable to various attacks and censorship attempts
- Tor is a constantly evolving network protocol to resists them

Backup slides

Tor: TLS Handshake (v1)

- Goal: Authenticate and establish TLS connection with shared session keys
- Any problems here?

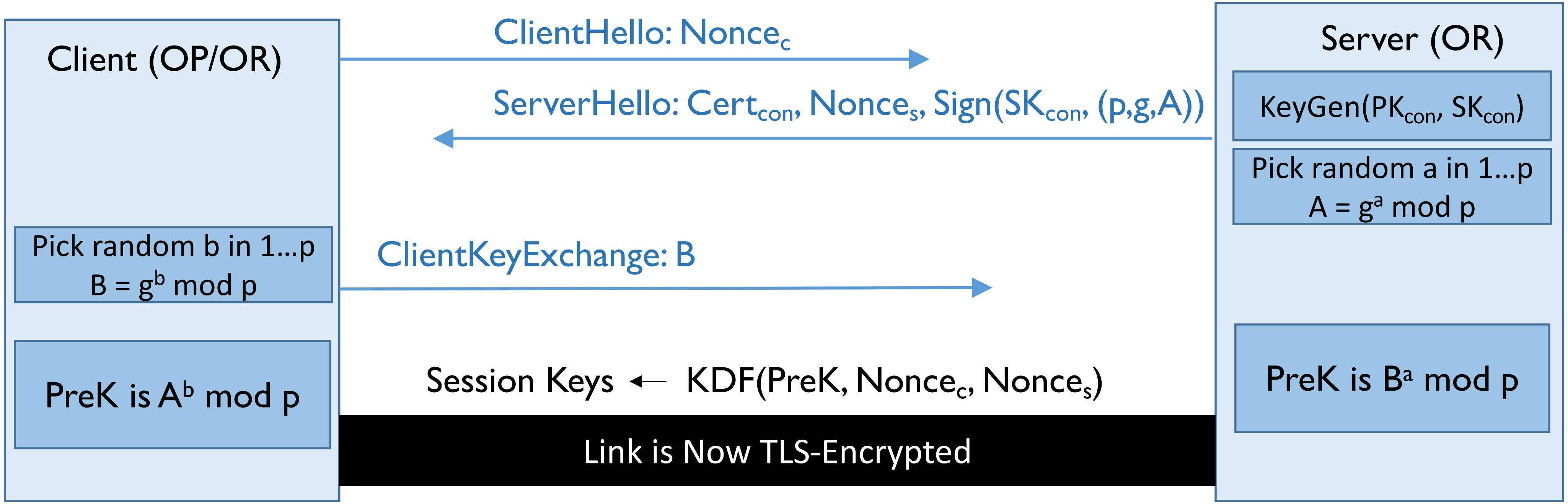


Client (OP/OR)
Validate p, g, A with PK_{con} in $Cert_{con}$
Authenticate Server with PK_{sid} in $Cert_{sid}$
Pick random b in 1…p
$B = g^b \bmod p$
PreK is $A^b \bmod p$
ClientHello: $Nonce_c$
ServerHello: $[Cert_{con}, Cert_{sid}]$, $Nonce_s$, $Sign(SK_{con}, (p, g, A))$
ClientAuth: $[Cert_{scon}, Cert_{cid}]$
ClientKeyExchange: B
Session Keys ← $KDF(PreK_s, Nonce_c, Nonce_s)$
Server (OR)
KeyGen (PK_{con}, SK_{con})
Pick random a in 1…p
$A = g^a \bmod p$
PreK is $B^a \bmod p$

Can enable fingerprinting attacks or censorship!

TOR: TLS Handshake

- First, establish TLS connection (looks like regular TLS handshake traffic)
- Then, do authentication "in-protocol" using Tor cells



Client (OP/OR)

Server (OR)

KeyGen(PK_{con}, SK_{con})

Pick random a in 1…p
A = g^a mod p

ClientHello: $Nonce_c$

ServerHello: $Cert_{con}$, $Nonce_s$, Sign(SK_{con}, (p,g,A))

Pick random b in 1…p
B = g^b mod p

ClientKeyExchange: B

PreK is A^b mod p

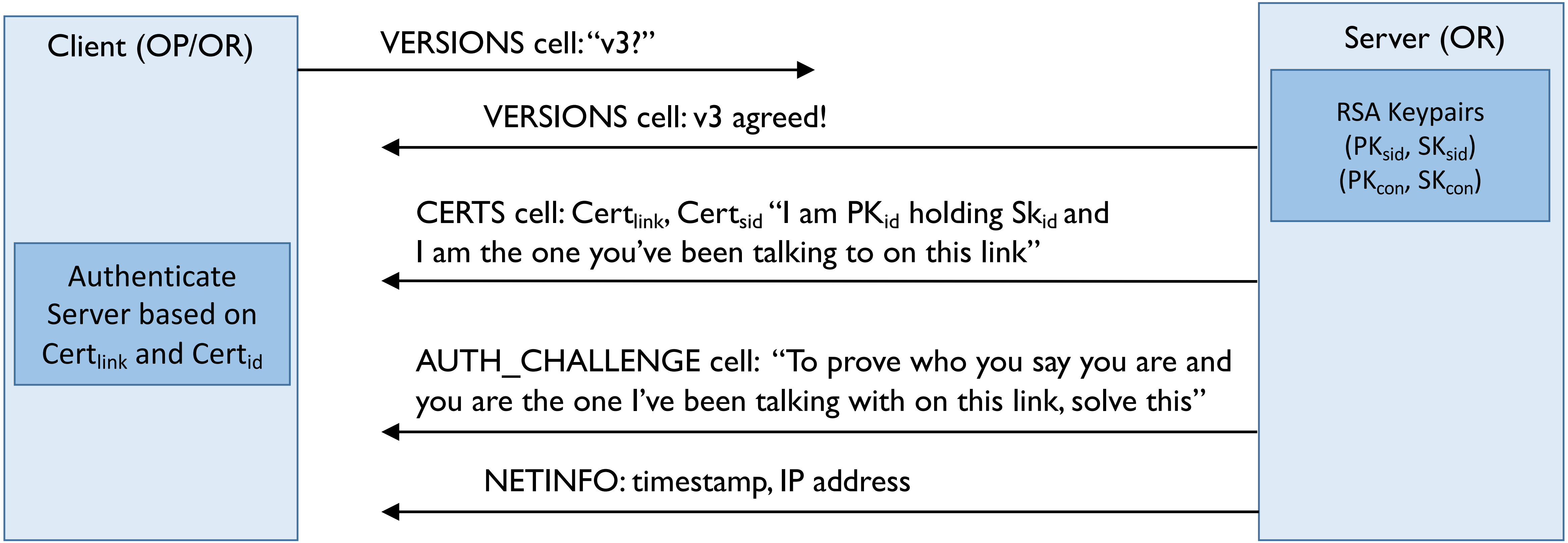
PreK is B^a mod p

Session Keys ← KDF(PreK, $Nonce_c$, $Nonce_s$)

Link is Now TLS-Encrypted

TOR: TLS Handshake

- Step 2: Authenticate Server using TOR cells



Client (OP/OR)

Server (OR)

RSA Keypairs
(PK_{sid}, SK_{sid})
(PK_{con}, SK_{con})

VERSIONS cell: "v3?"

VERSIONS cell: v3 agreed!

CERTS cell: $Cert_{link}$, $Cert_{sid}$ "I am PK_{id} holding Sk_{id} and I am the one you've been talking to on this link"

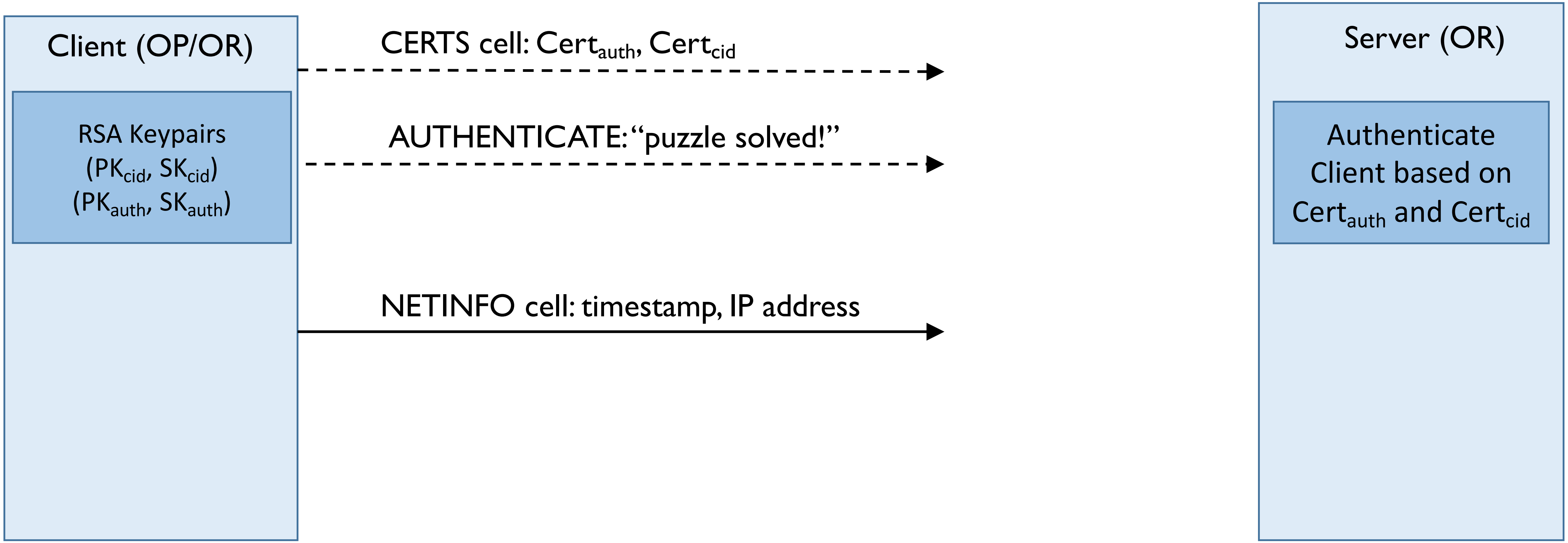
AUTH_CHALLENGE cell: "To prove who you say you are and you are the one I've been talking with on this link, solve this"

NETINFO: timestamp, IP address

Authenticate Server based on $Cert_{link}$ and $Cert_{id}$

TOR: TLS Handshake

- Step 3: Client authentication (optional) and client network info shared



Client (OP/OR)

RSA Keypairs
(PK_{cid}, SK_{cid})
(PK_{auth}, SK_{auth})

CERTS cell: $Cert_{auth}$, $Cert_{cid}$

AUTHENTICATE: "puzzle solved!"

NETINFO cell: timestamp, IP address

Server (OR)

Authenticate Client based on $Cert_{auth}$ and $Cert_{cid}$