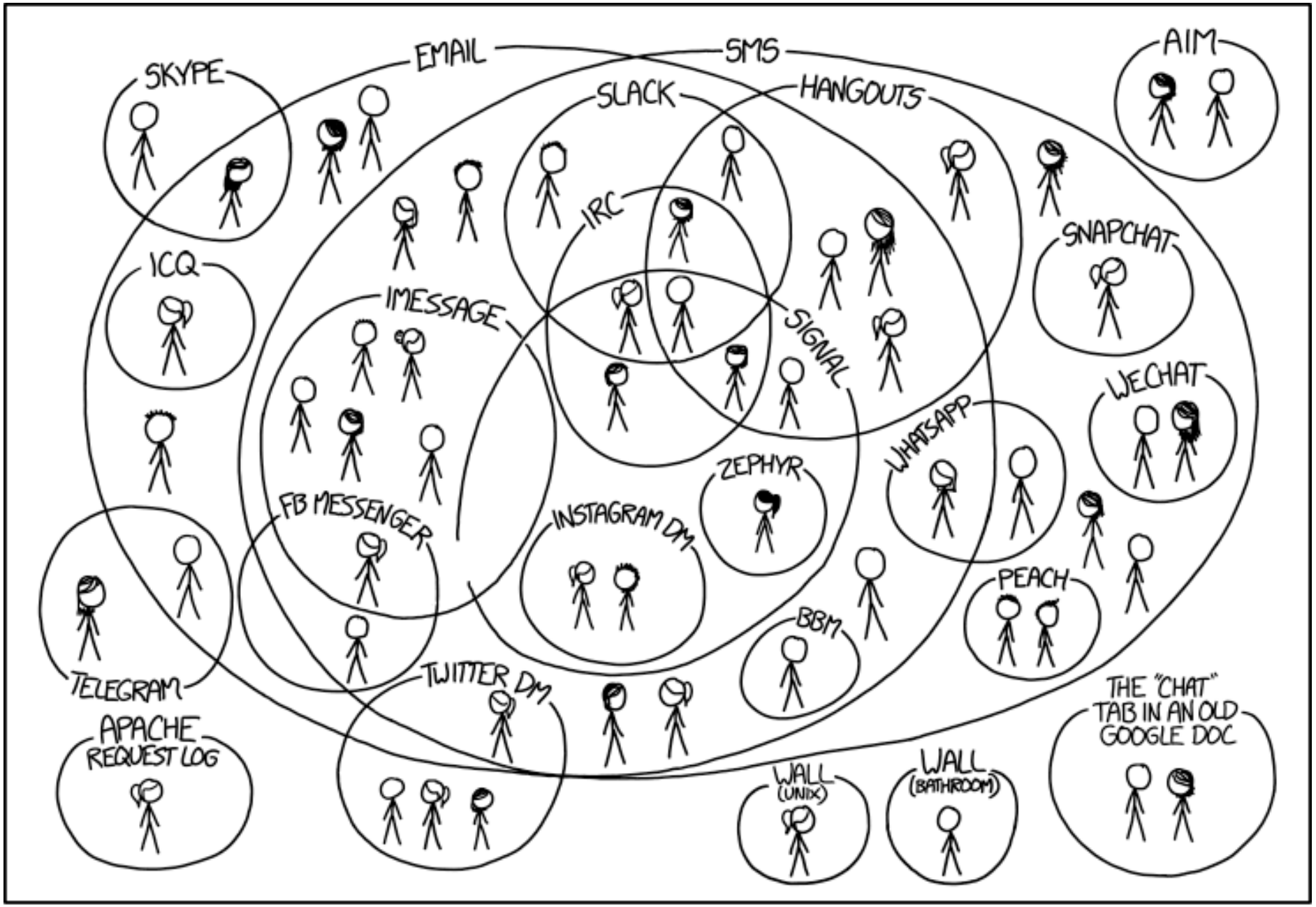
CS 55:
Security and Privacy

Secure comms



I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.

https://xkcd.com/1810/

Do not forget about physical security

https://www.youtube.com/watch?v=nJu_-Iuppc0&start=21

Discussion

What properties would you like in “secure” communications?

Agenda

1. The Onion Router (TOR)
2. Transport Layer Security (TLS)
3. Virtual Private Networks (VPNs)
4. Signal/WhatsApp

Discussion

What is the difference between security and anonymity?

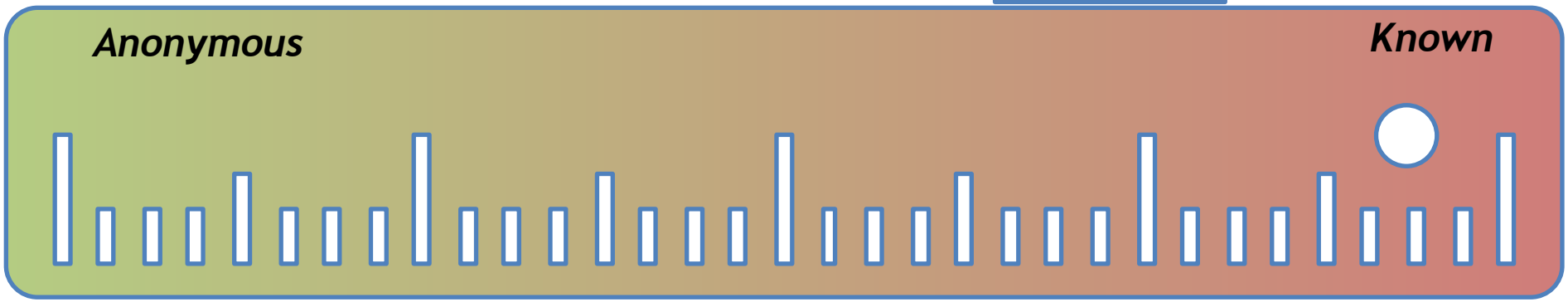
Who needs anonymity?

Sometimes you don't want anyone to know you've sent a message or to whom

cannot tell when a message is sent or not sent

non-trivial probability that the sender is not the node in question

sender whose communications can be identified is exposed.

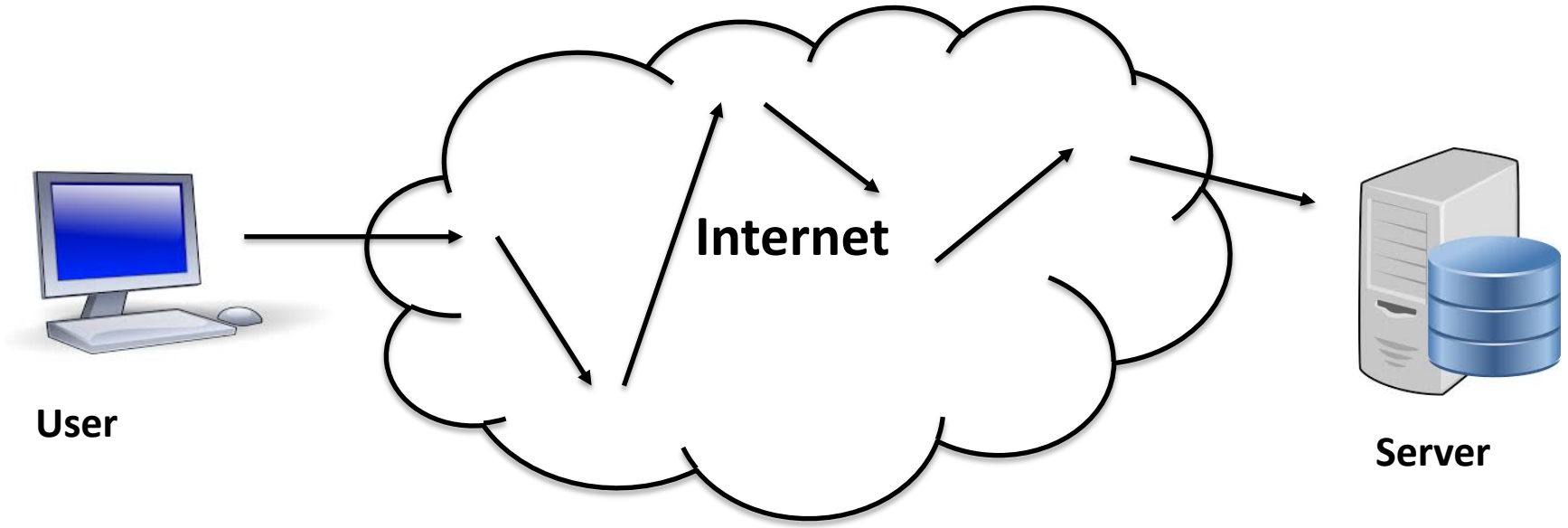


can see that a message is sent but cannot be sure where it came from

sender is probably innocent if the sender is no more likely than not to be the originator of a message

can demonstrate this exposure to other entities, the sender is provably exposed.

When you logon to a server, anyone sniffing along the way can see the traffic



The Onion Router (TOR) obscures a message *and* the source and destination

User

User wants to access server, but doesn't want anyone to know they are accessing that server

TOR Node | TOR Node | TOR Node | TOR Node

TOR Node | TOR Node | TOR Node | TOR Node

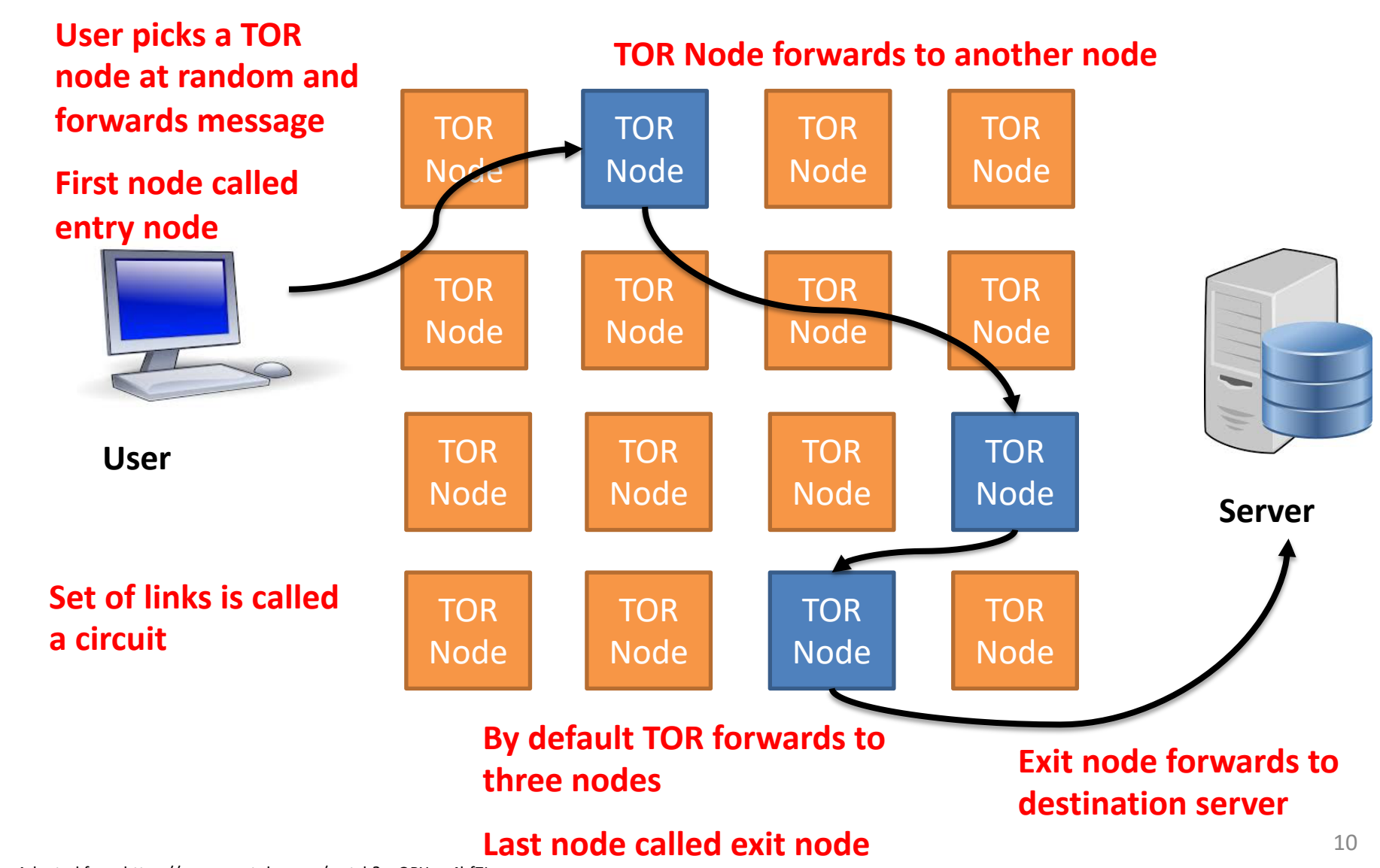
TOR Node | TOR Node | TOR Node | TOR Node

TOR Node | TOR Node | TOR Node | TOR Node

Server

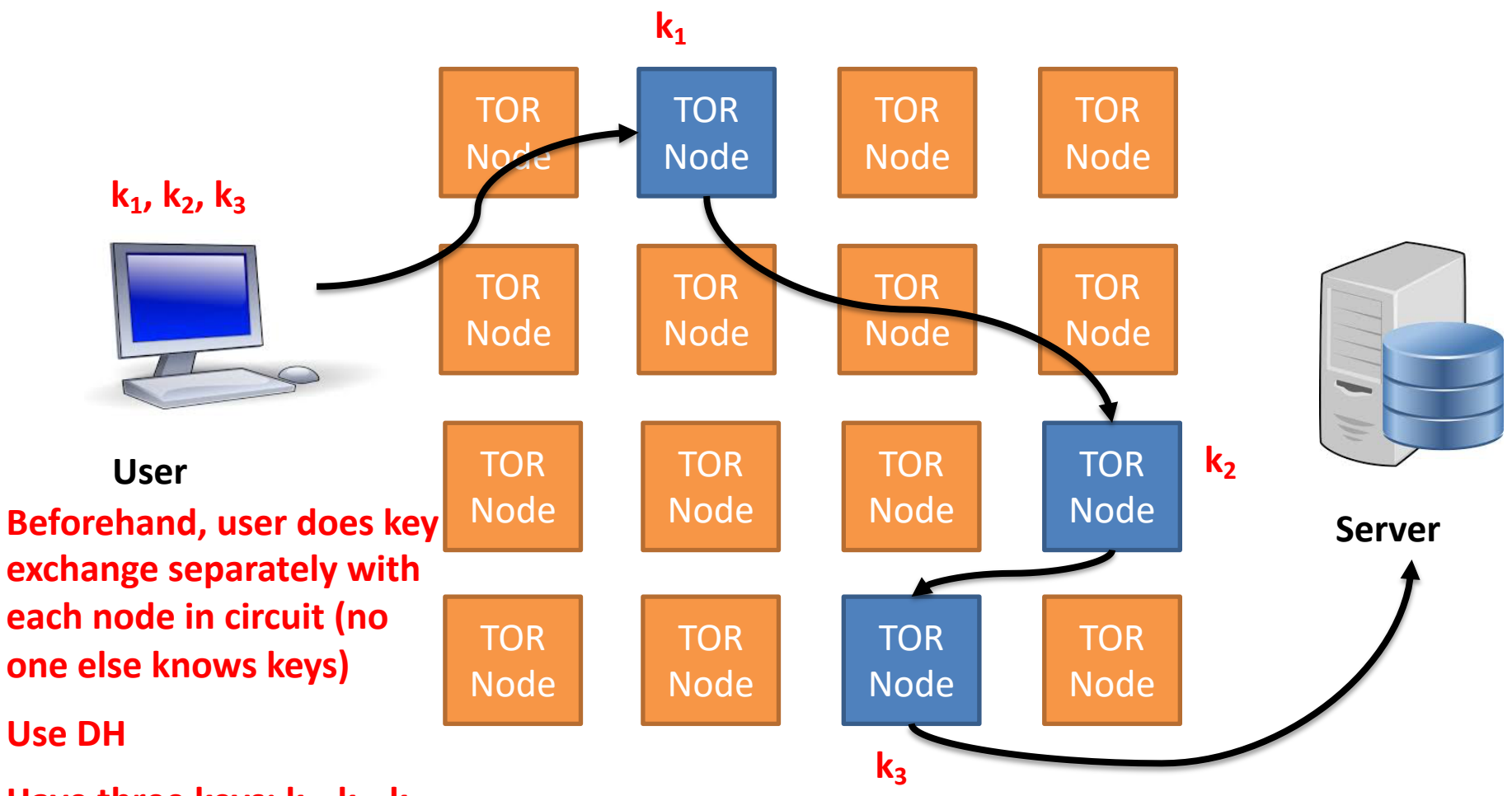
Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



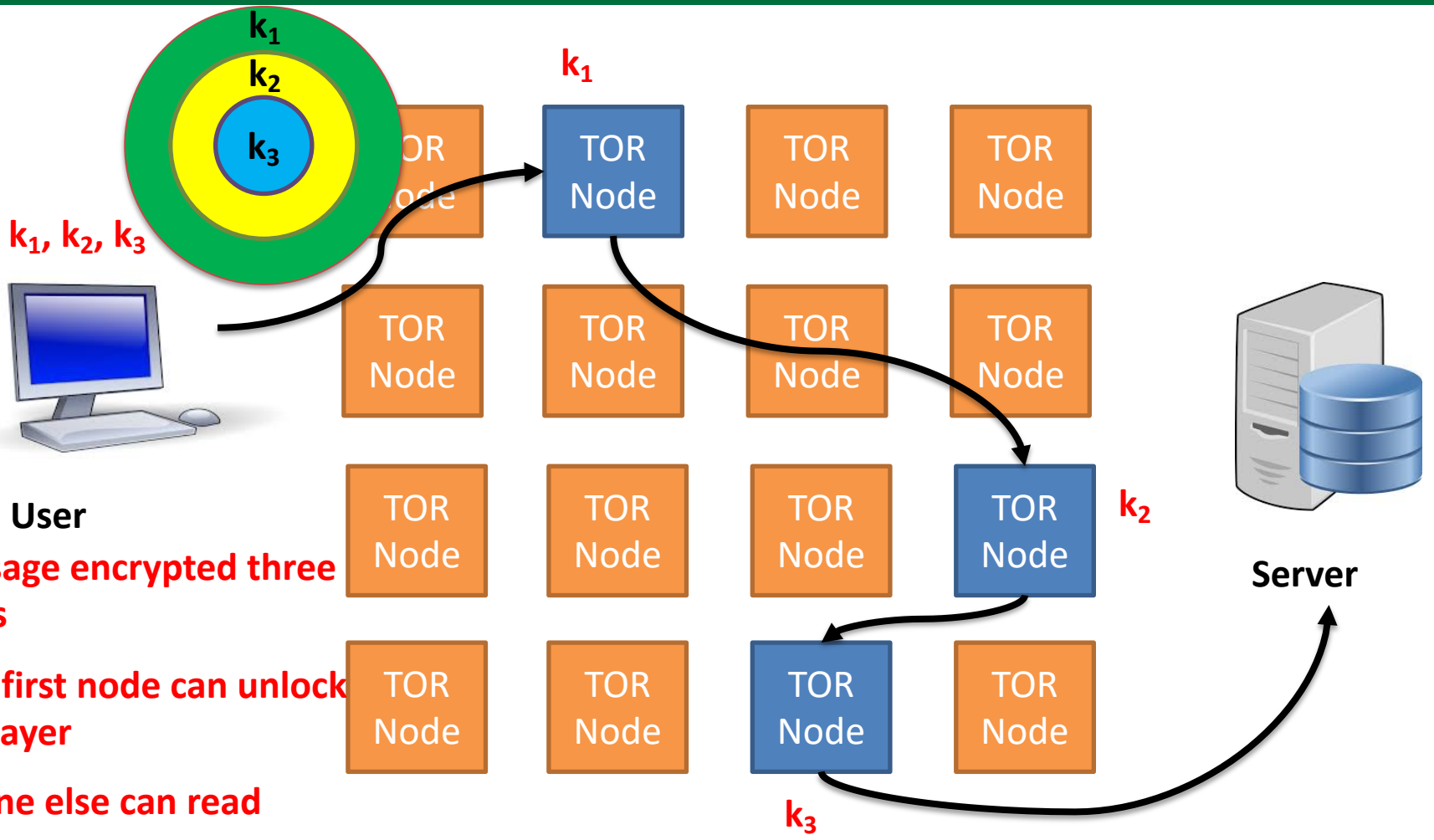
Beforehand, user does key exchange separately with each node in circuit (no one else knows keys)

Use DH

Have three keys: k_1, k_2, k_3

Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



Message encrypted three times

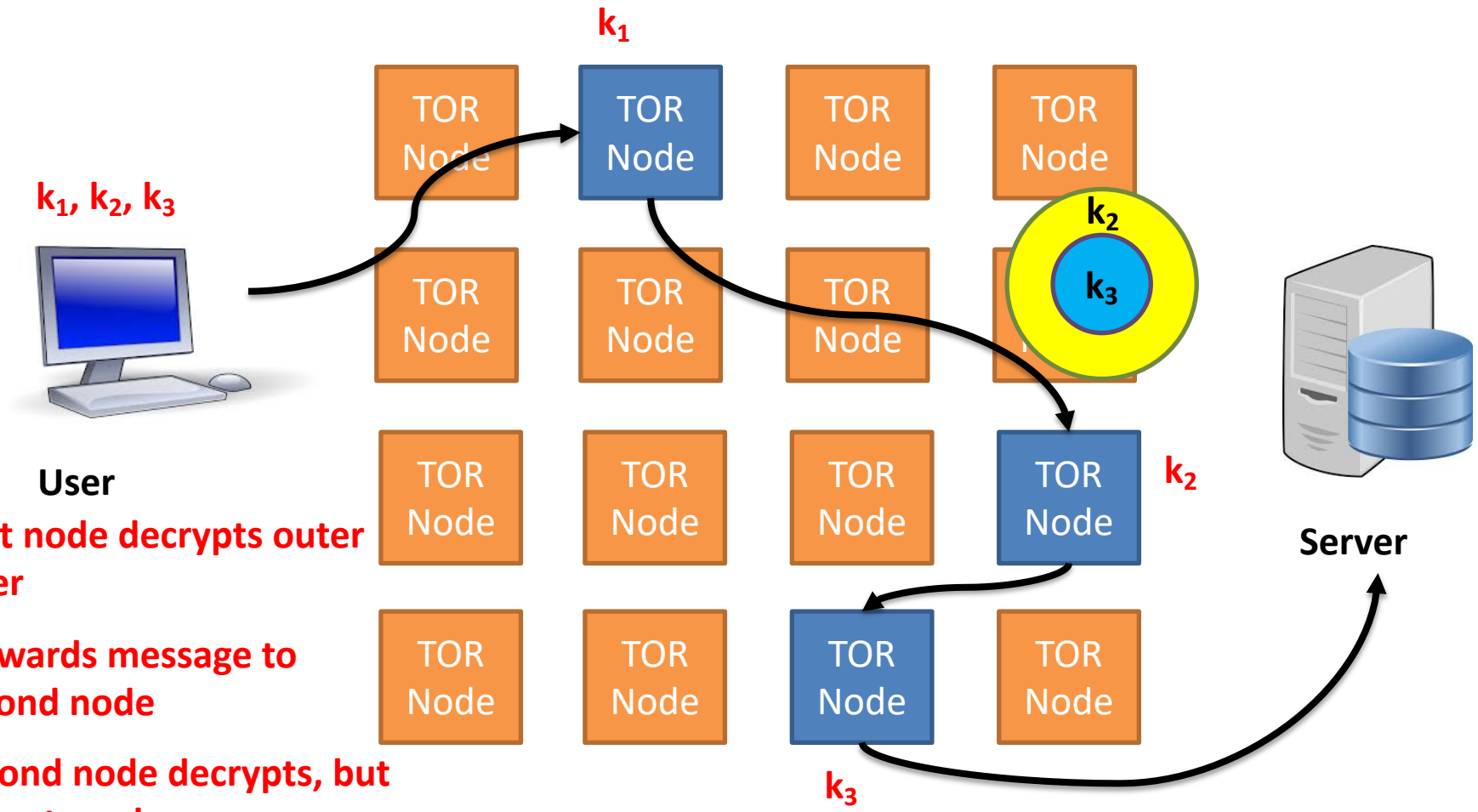
Only first node can unlock first layer

No one else can read

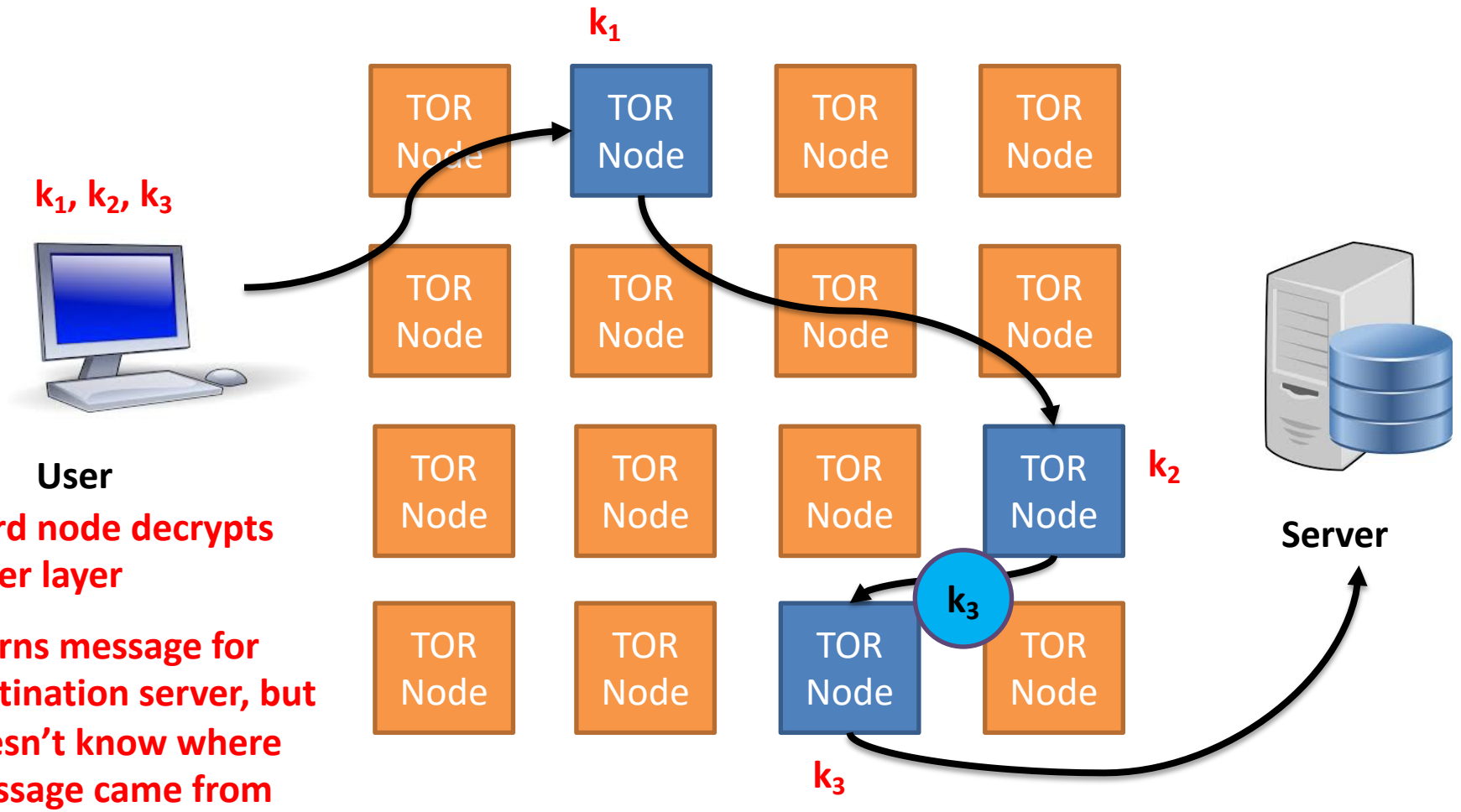
First node cannot read message or destination

Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



The Onion Router (TOR) obscures a message *and* the source and destination

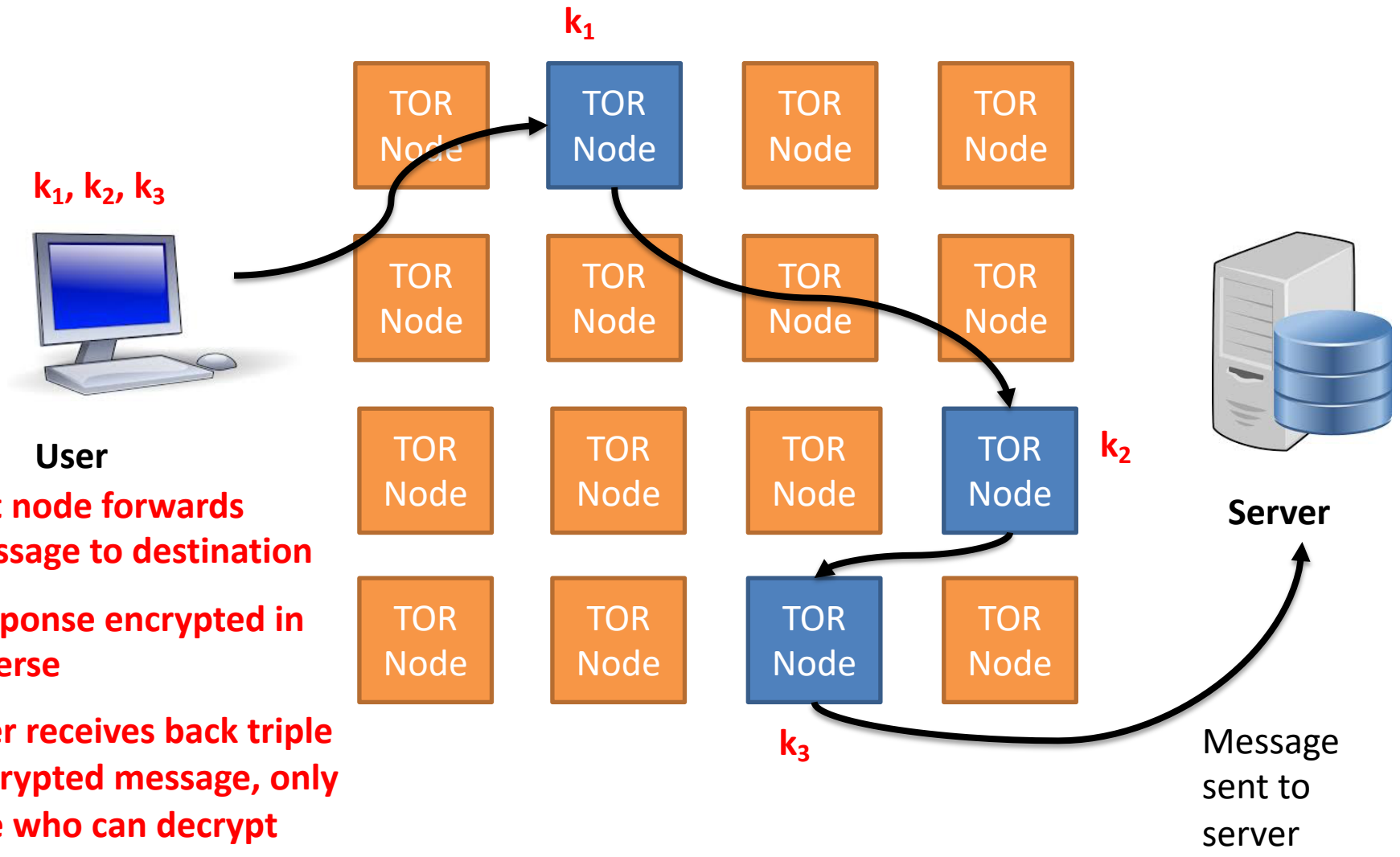


Third node decrypts outer layer

Learns message for destination server, but doesn't know where message came from originally (does know previous node)

Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



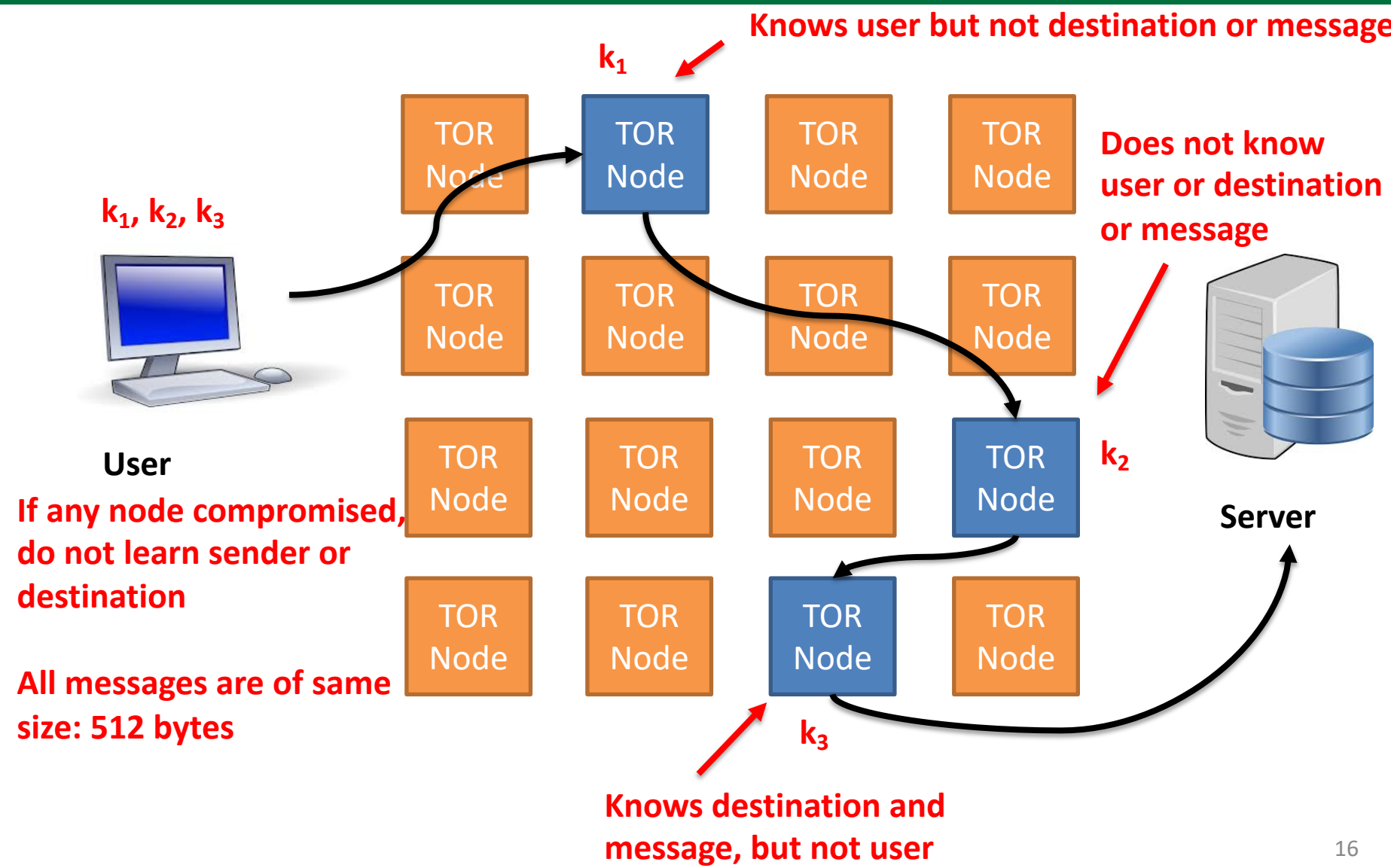
Exit node forwards message to destination

Response encrypted in reverse

User receives back triple encrypted message, only one who can decrypt

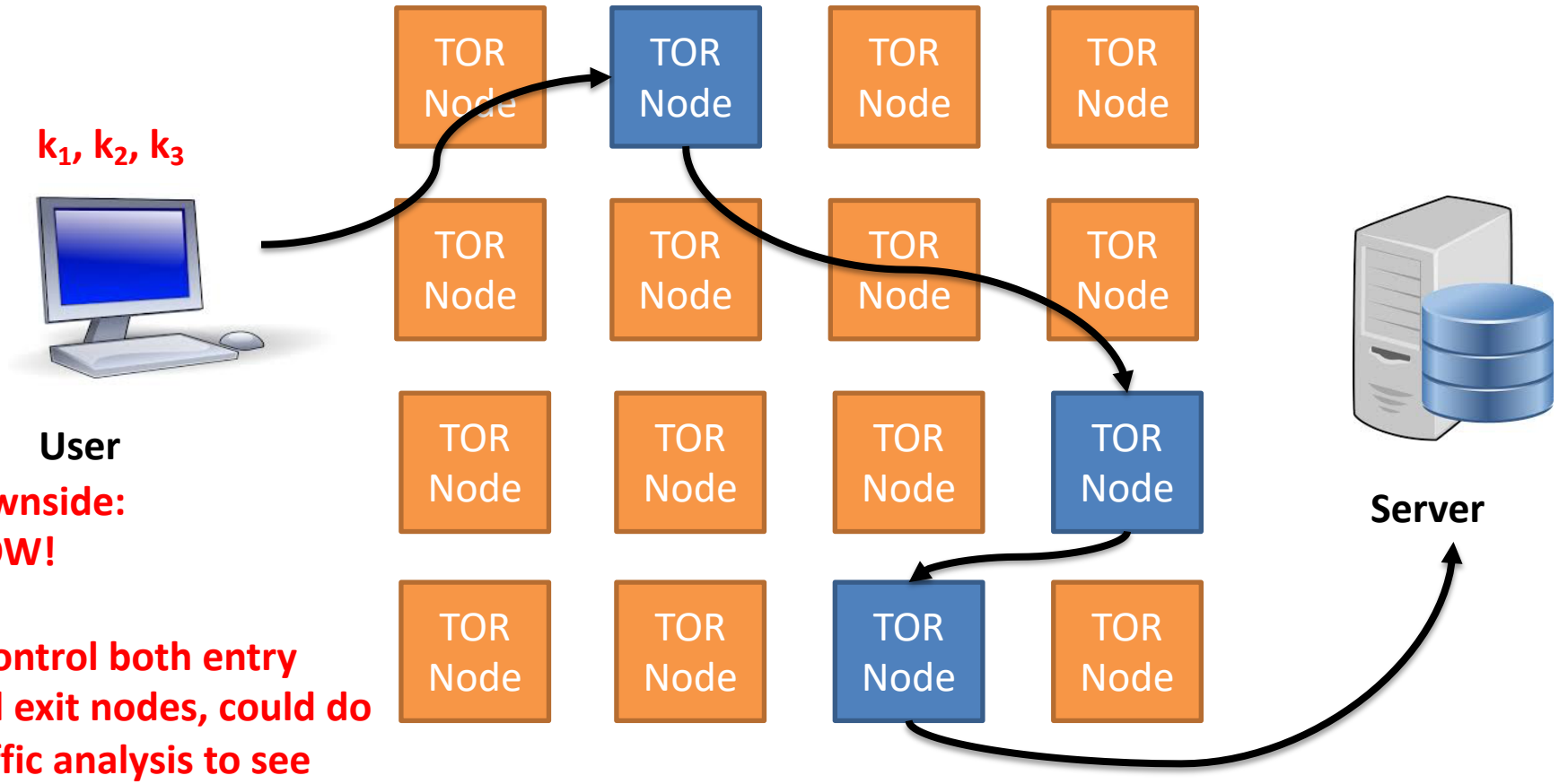
Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

The Onion Router (TOR) obscures a message *and* the source and destination



Downside: SLOW!

If control both entry and exit nodes, could do traffic analysis to see who is talking to whom

Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

DEMO

Start TOR browser

Go to www.amazon.com (will be slow to load, but Amazon doesn't know its me, but show me in Poland or else where)

See location by visiting https://ipapi.co/

Agenda

1. The Onion Router (TOR)
2. Transport Layer Security (TLS)
3. Virtual Private Networks (VPNs)
4. Signal/WhatsApp

Transport Layer Security (TLS) provides a secure channel between two parties

The secure channel has 3 properties:

1. **Confidentiality**: Nobody other than the two ends of the channel can see the actual content of the data transmitted
2. **Integrity**: Channel can detect any changes made to the data during transmission
3. **Authentication**: At least one end of the channel needs to be authenticated, so the other end knows with whom it is talking

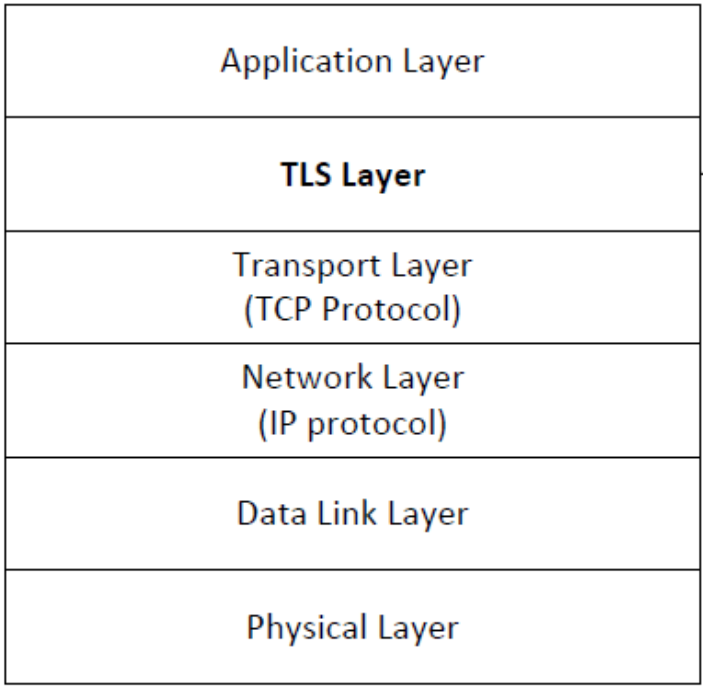
TLS normally done between a client and a server (e.g., web browser and web server)

TLS grew out of SSL

- **You will often hear people say SSL when they mean TLS**
- **You'll sometimes see SSL/TLS**

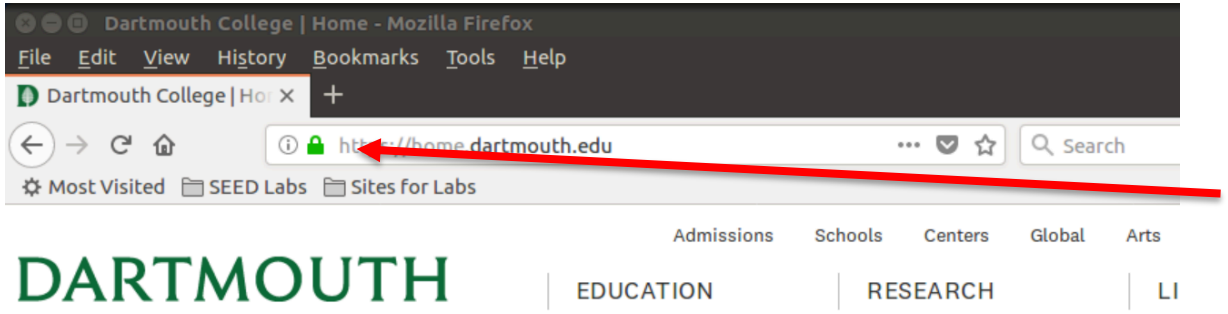
TLS sits between the Transport and Application layers

- Unprotected data is given to TLS by Application layer
- TLS handles encryption, decryption and integrity checks
- TLS gives protected data to Transport layer



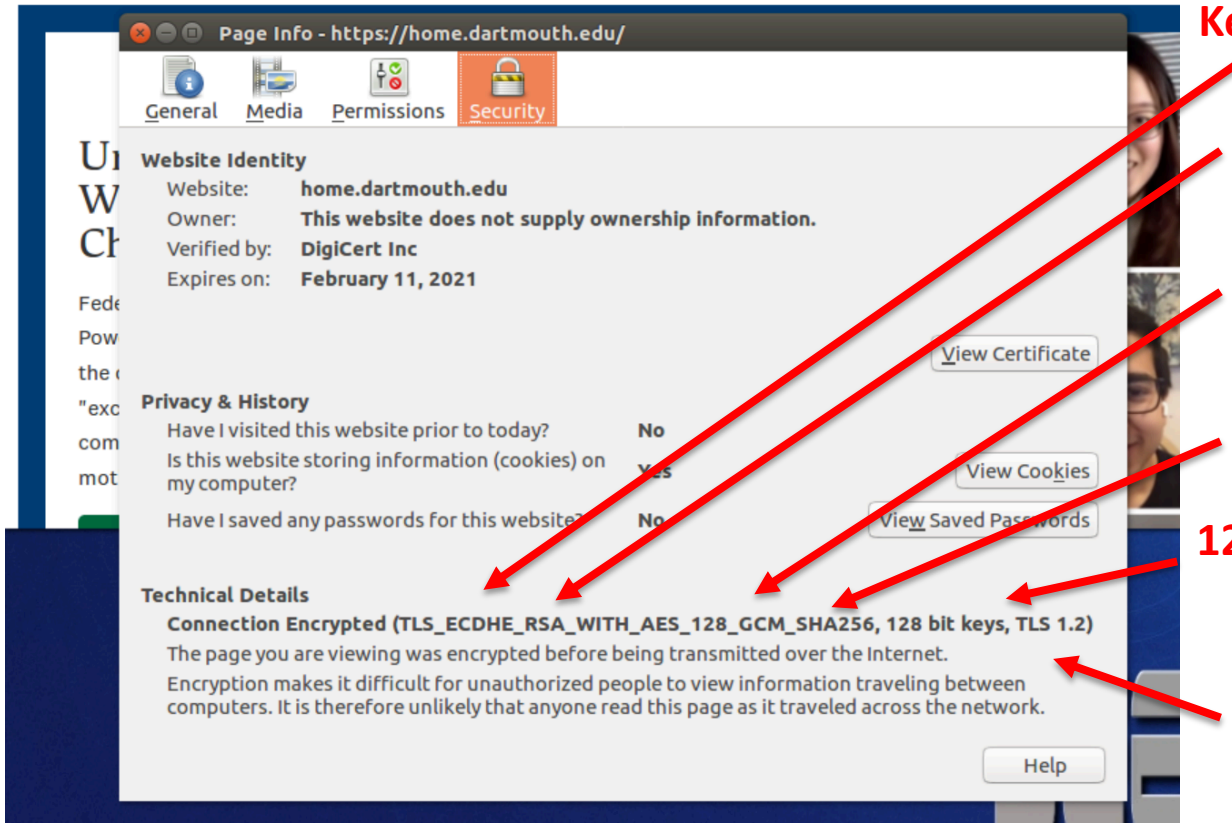
Adapted from *Computer & Internet Security* by Du

You can see the details in your browser



In Firefox

- Click on lock in URL bar
- Select More Information



Key exchange uses ECDHE

RSA for public key authentication of certificates

128-bit AES encryption using GCM

SHA256 for hashing

128-bit keys

TLS version 1.2

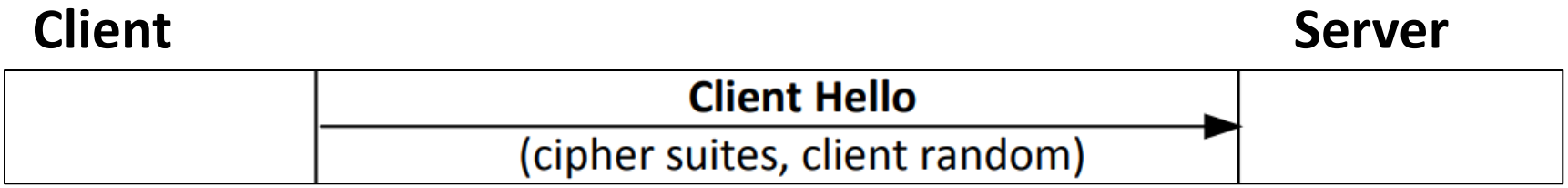
TLS involves a handshake between client and server to agree upon parameters

Before a client and server can communicate securely, several things need to be set up first:

- Encryption algorithm and key
- MAC algorithm
- Algorithm for key exchange

These cryptographic parameters need to be agreed upon by both the client and the server, otherwise connection is refused

TLS involves a handshake between client and server to agree upon parameters

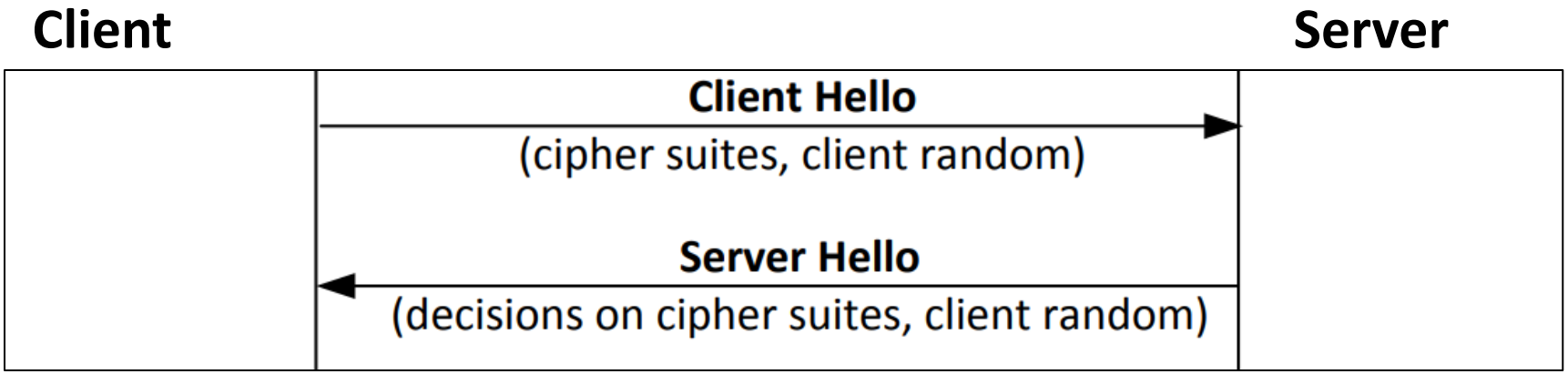


Client sends "Client Hello" message to server with:

- **List of ciphers that it can use (e.g., AES)**
- **Random nonce (to prevent replay attacks)**
- **Max TLS version it can support (e.g., TLS 1.2, 1.3)**

Adapted from https://www.youtube.com/watch?v=86cQJ0MMses

TLS involves a handshake between client and server to agree upon parameters

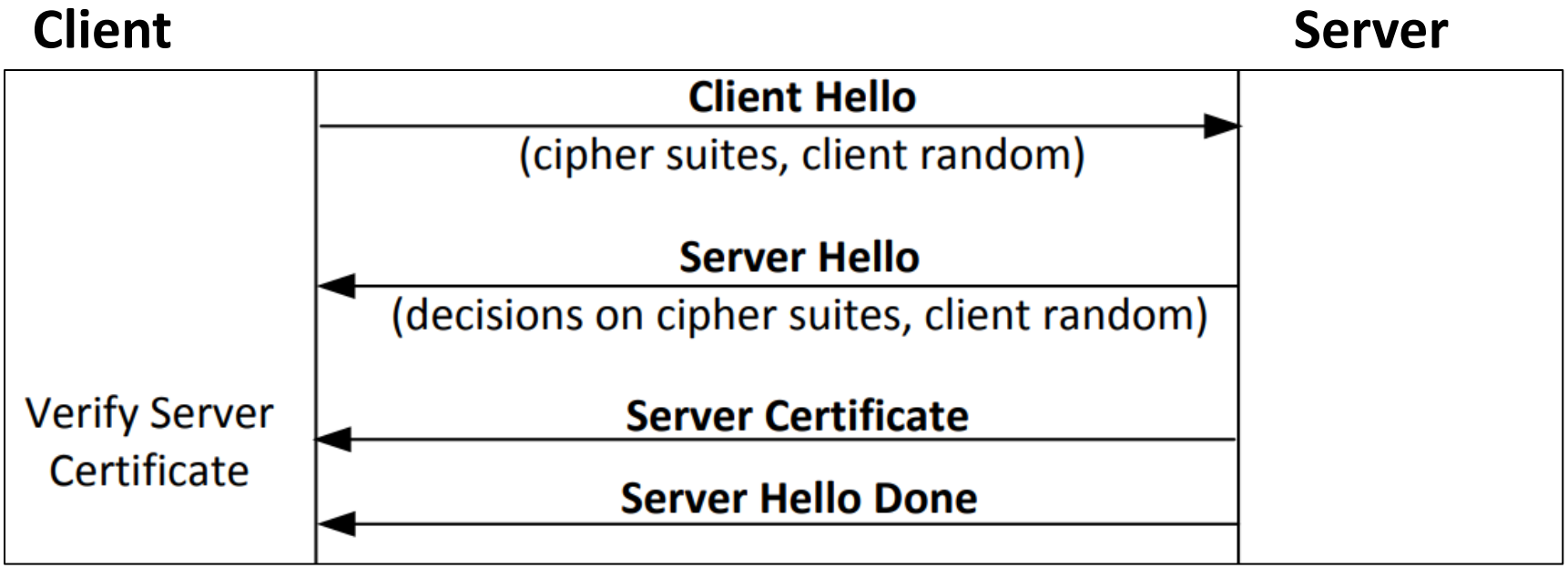


Server responds with "Server Hello" message to client with:

- **A decision on what cipher to use**
- **Random nonce (to prevent replay attacks)**
- **TLS version to use**

Adapted from https://www.youtube.com/watch?v=86cQJ0MMses

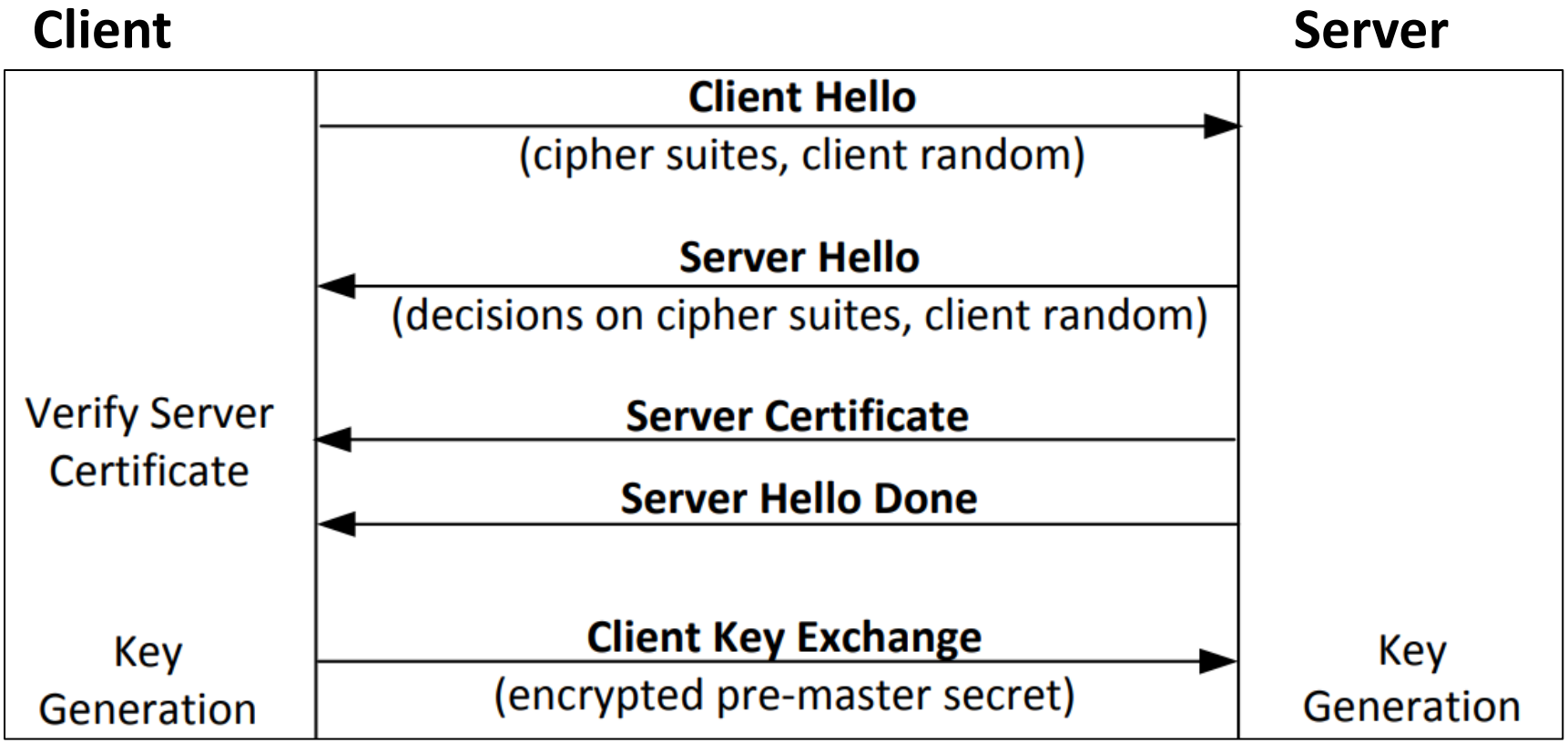
TLS involves a handshake between client and server to agree upon parameters



Server sends its certificate (includes public key)
Client verifies certificate (going up to root if needed)
Client now knows server is the intended server
Hello Done indicates the first portion of handshake is complete
In some use cases, the client also sends a certification to server (typically not done on the web)

Adapted from https://www.youtube.com/watch?v=86cQJ0MMses

TLS involves a handshake between client and server to agree upon parameters

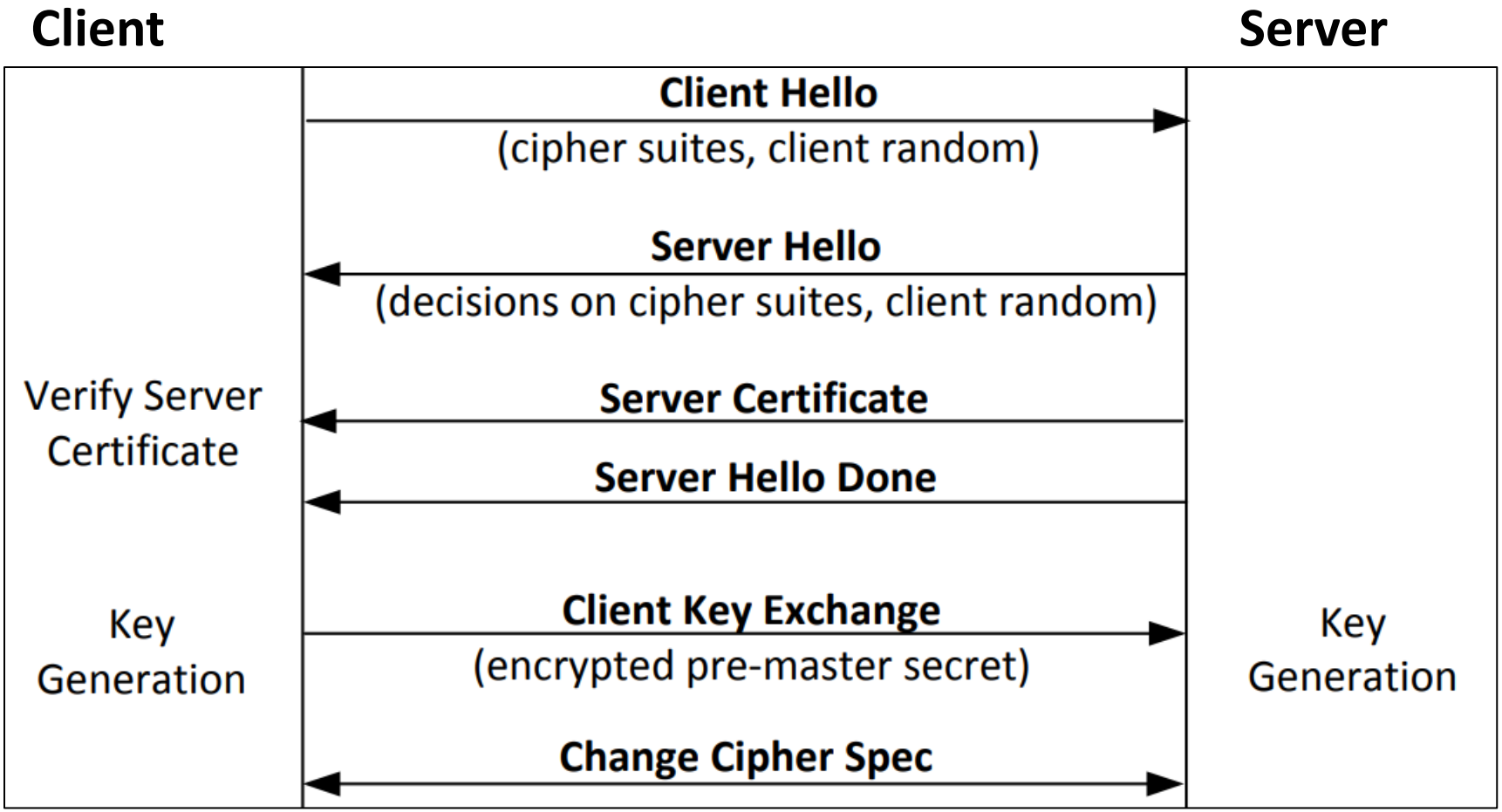


Client creates pre-master secret

- **Produces a random number to serve as the pre-master secret**
- **Encrypts random number with server's public key and sends to server**

Adapted from https://www.youtube.com/watch?v=86cQJ0MMses

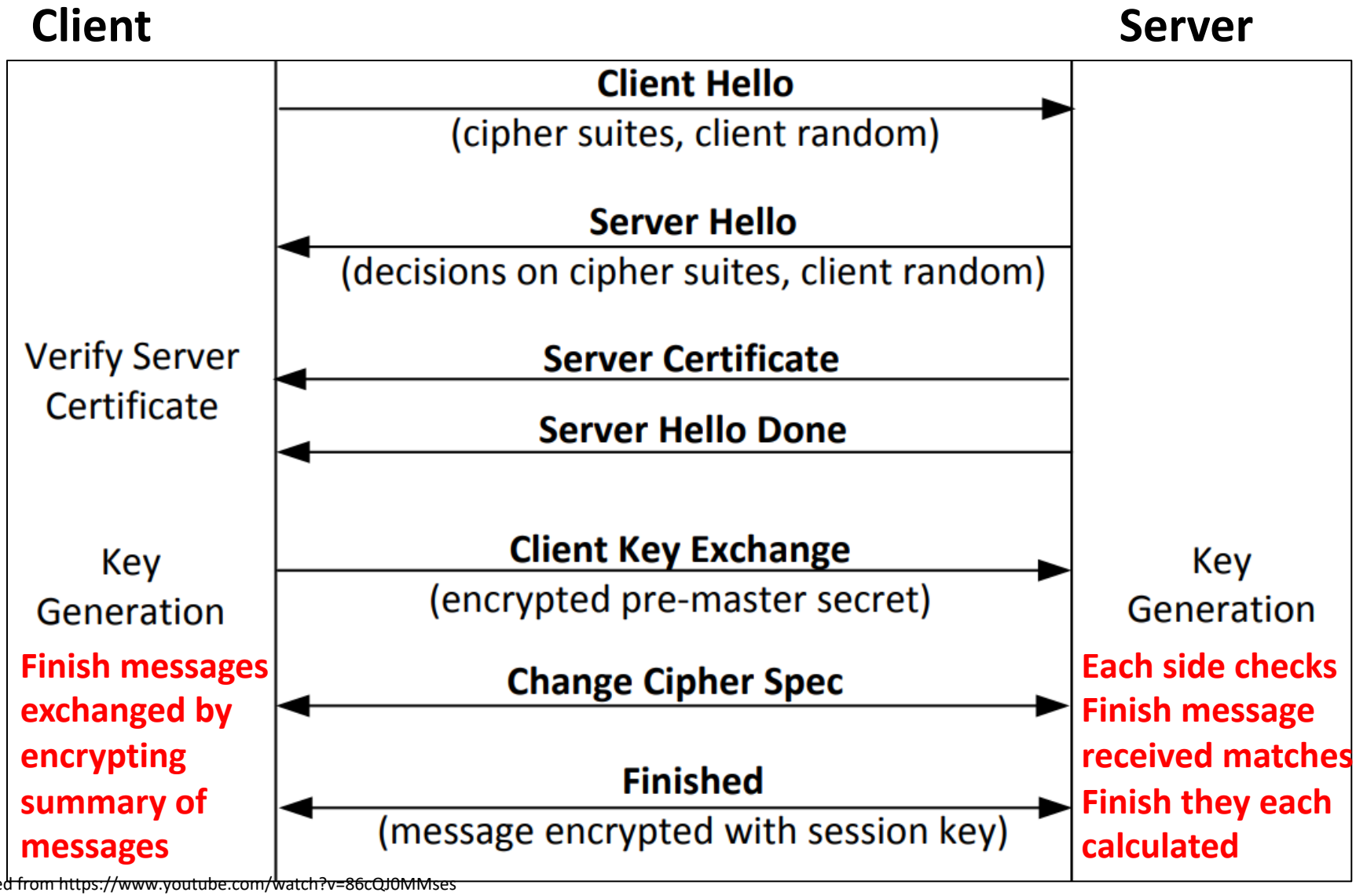
TLS involves a handshake between client and server to agree upon parameters



Server and client create 48-byte master secret by combining pre-master secret with nonces

Adapted from https://www.youtube.com/watch?v=86cQJ0MMses

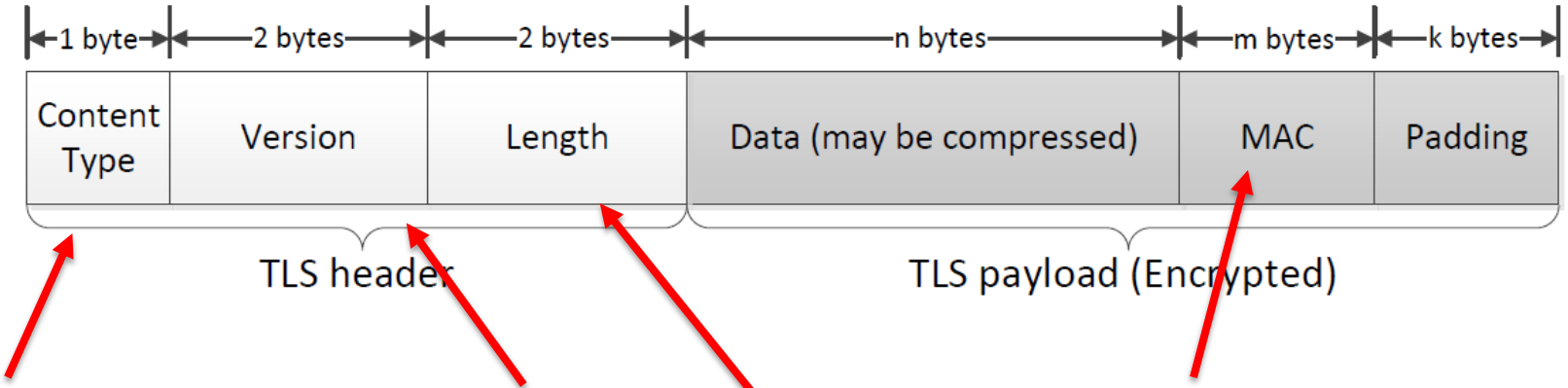
TLS involves a handshake between client and server to agree upon parameters



Adapted from https://www.youtube.com/watch?v=86cQJ0MMses

After handshake, client and server exchange encrypted data using records

Data is transferred using records, each record contains a header and a payload



Indicates the type of data carried

- **Alert**
- **Application**
- **Heartbeat**
- **Change Cipher Spec**

Which TLS version to use

- **1.2**
- **1.3**

How many bytes are in the payload

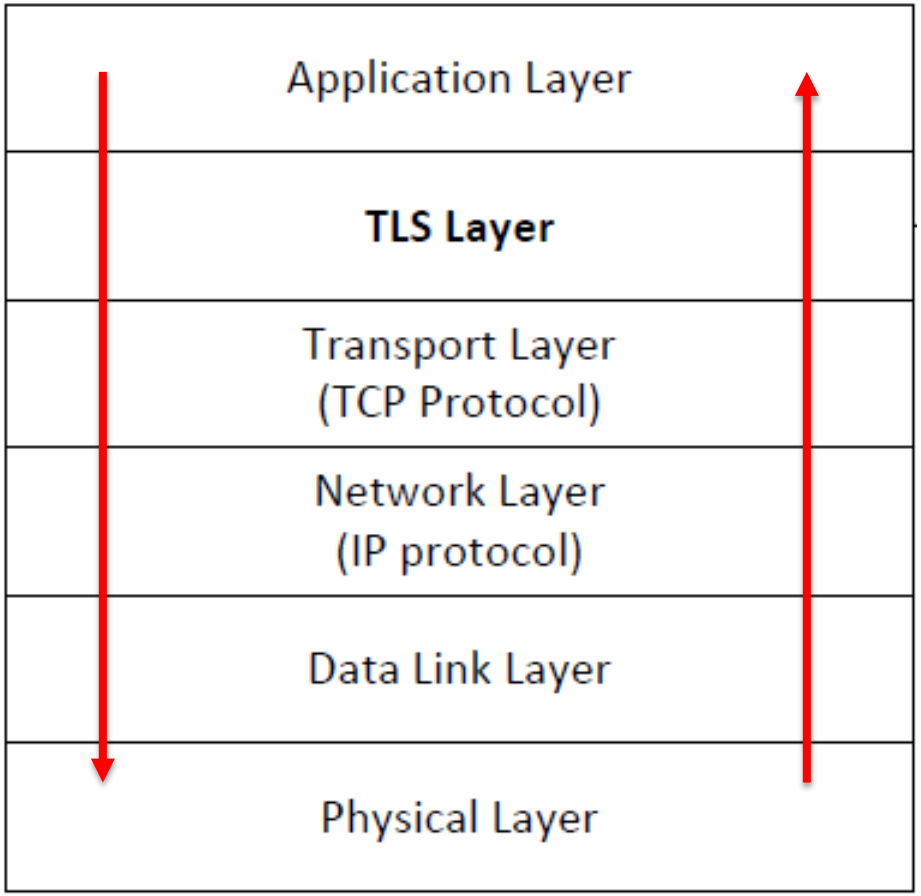
Message Authentication Code (MAC)

- **Sender computes based on data and sequence number**
- **Receiver computes MAC on data and checks with received MAC**
- **Detects if data modified**

Remember TLS sits between the Application and Transport Layers

Sender

- Takes data from Application Layer
- Adds header and encrypts payload
- Sends record to Transport Layer and down stack



Receiver

- Takes data from TCP Layer
- Strips header and decrypts
- Sends to Application Layer

Agenda

1. The Onion Router (TOR)
2. Transport Layer Security (TLS)
3. Virtual Private Networks (VPNs)
4. Signal/WhatsApp

Local Area Networks (LANs) are networks set up for a physical location

Often people outside want access to LAN

- Work from home
- Travelling
- Customer/partners

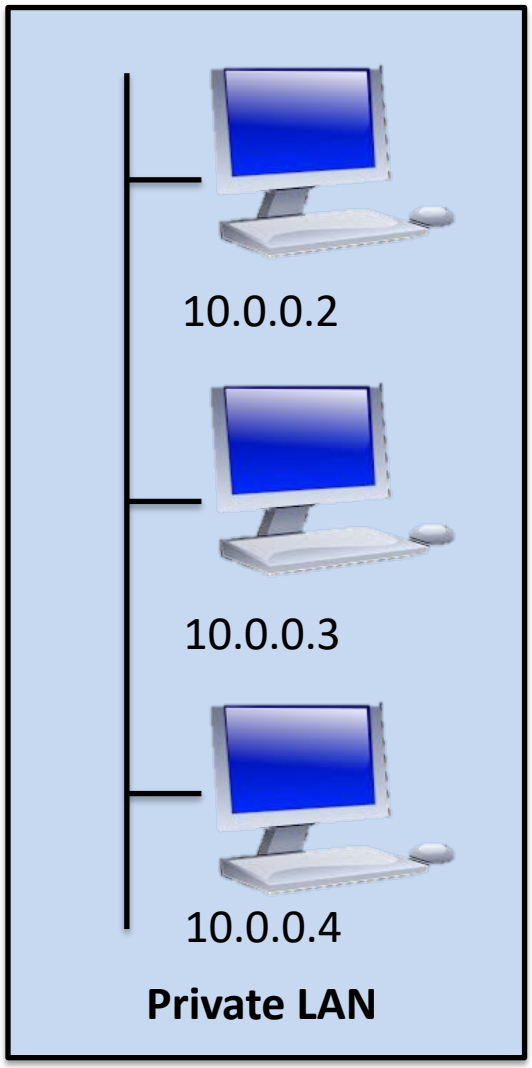
Could give devices routable IP address (or do port forwarding) and open them to the Internet

Problem?

- Increases attack surface!
- Netscout found devices attacked within 5 minutes
- Hard and crunchy on the outside, software and chewy on the inside!

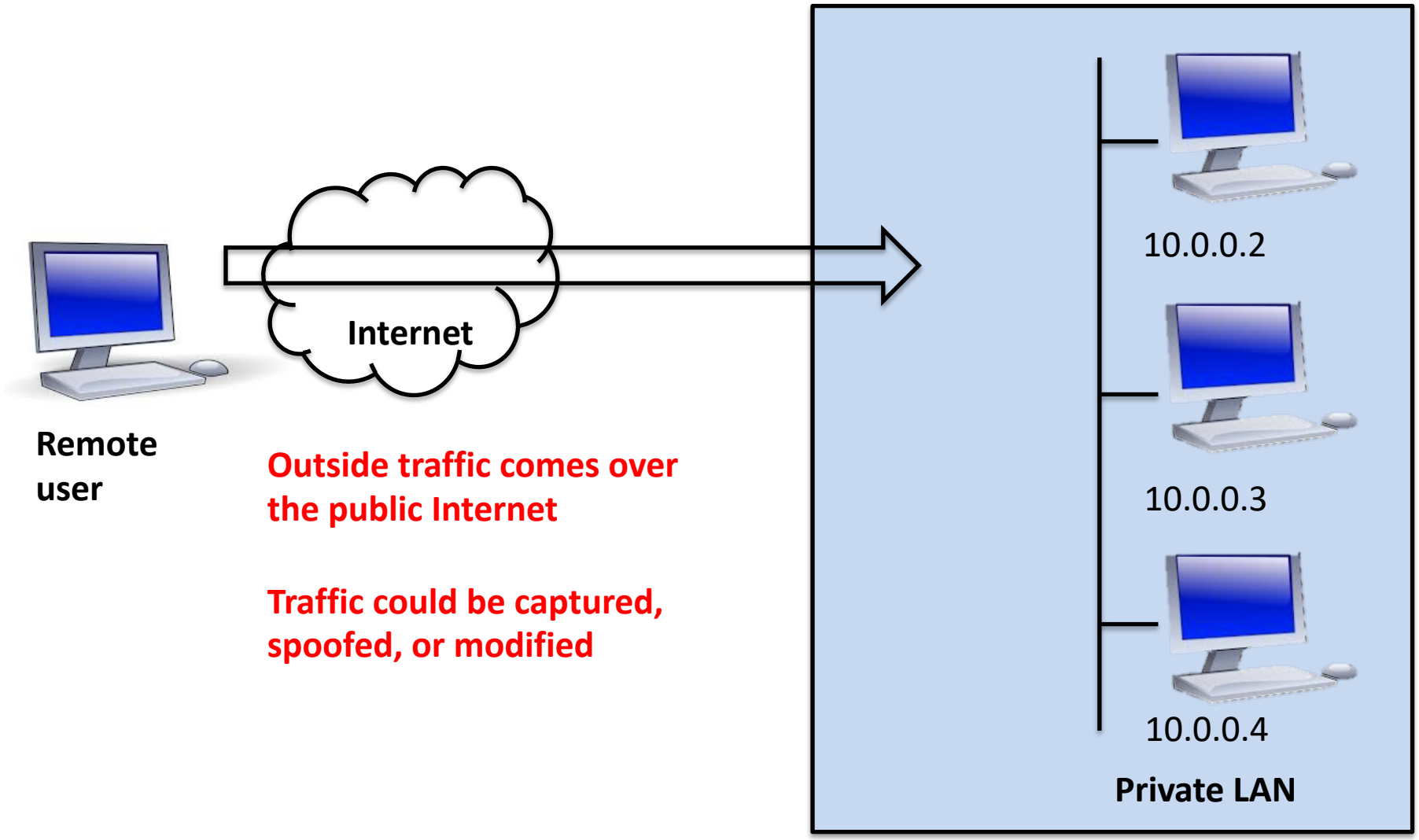
Devices set up with addresses in non-routable range(e.g., 10.0.0.0/8 or 192.168.0.0/16)

Device in local area able to talk to each other (router not shown)



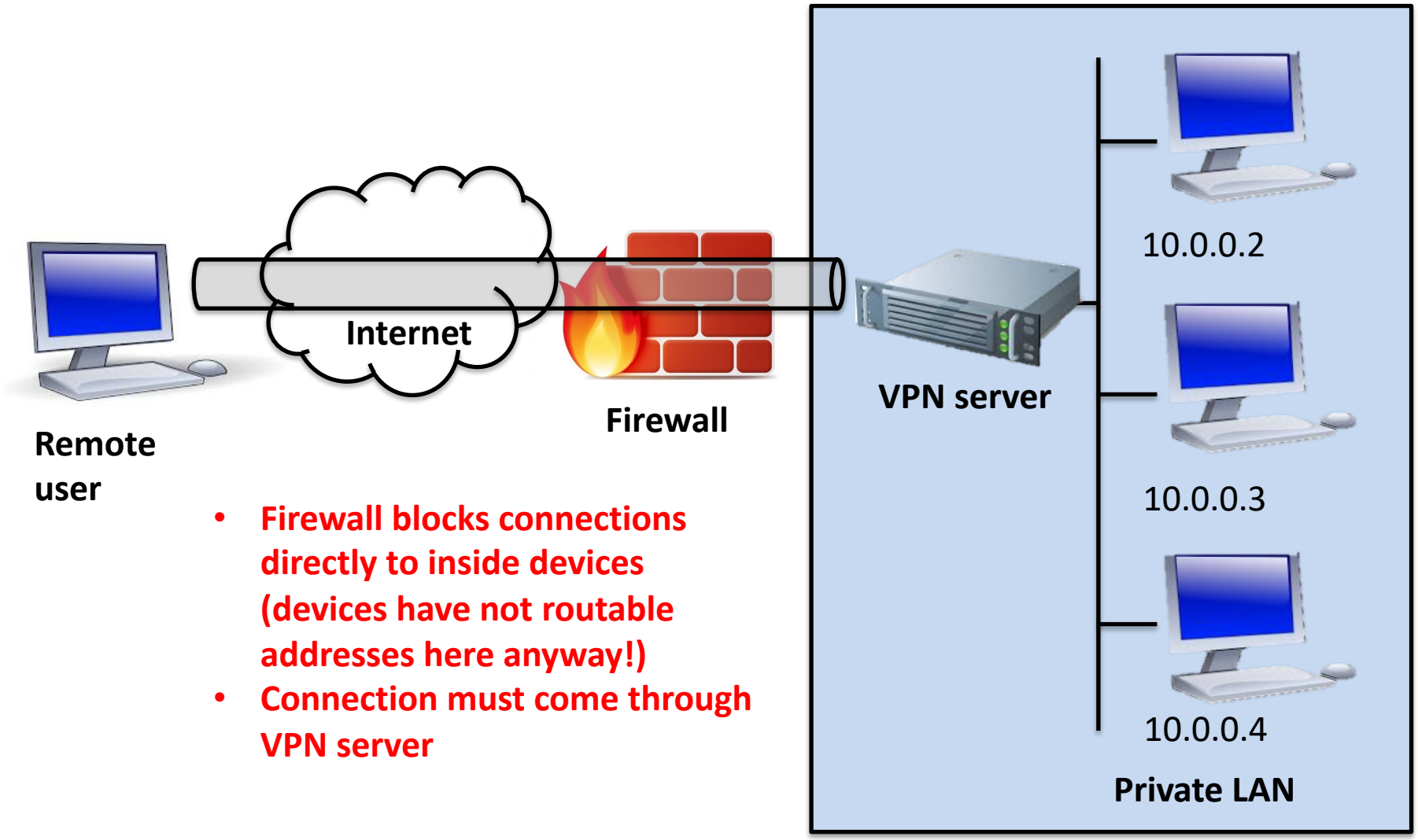
https://www.netscout.com/sites/default/files/2019-07/SECR_010_EN-1901%20%E2%80%93%20NETSCOUT%20Threat%20Report%201H%202019%20%E2%80%93%20Web.pdf

Outside traffic that comes over the public Internet cannot be trusted



Adapted from *Computer & Internet Security* by Du

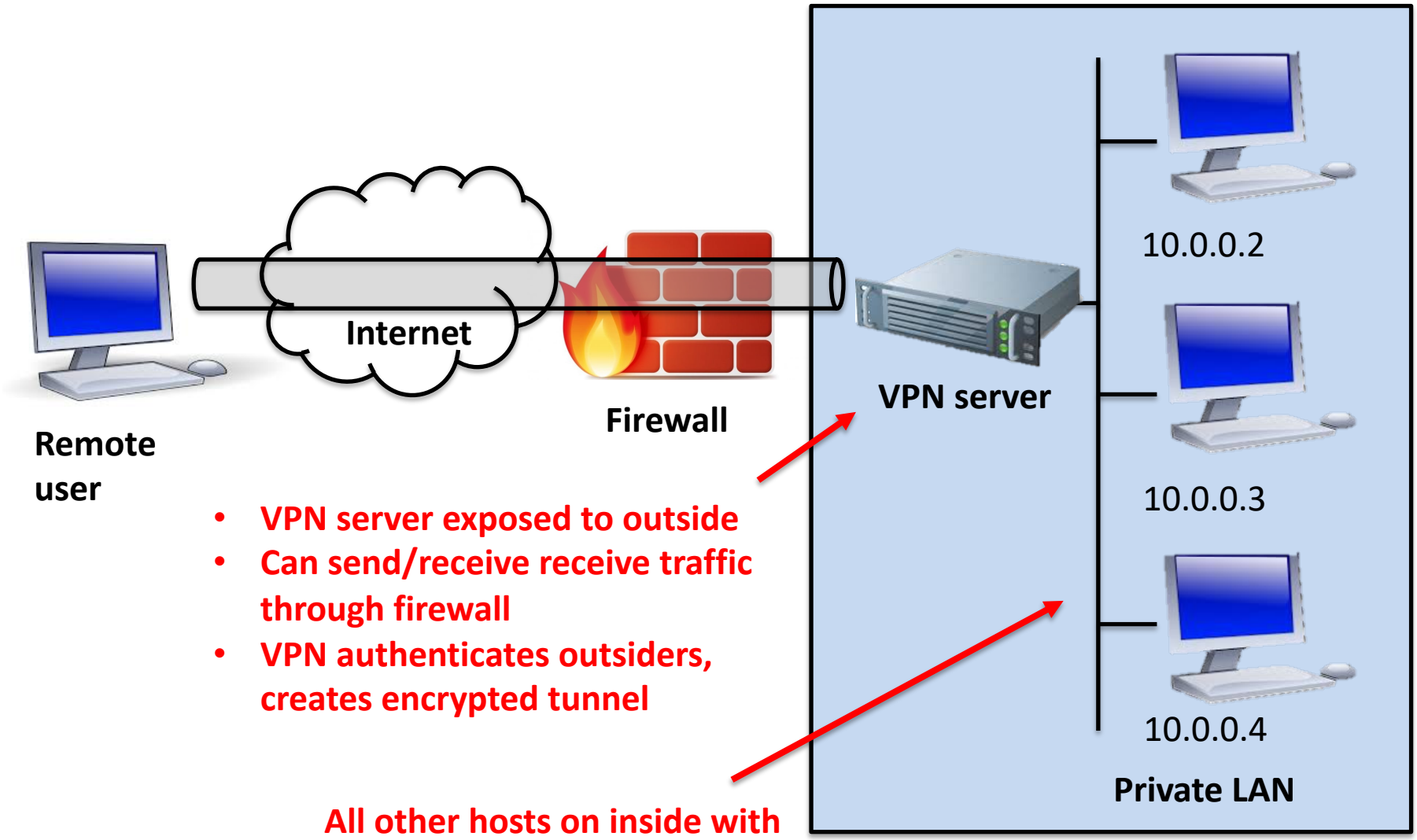
All traffic from outside to inside devices must come through VPN server



- **Firewall blocks connections directly to inside devices (devices have not routable addresses here anyway!)**
- **Connection must come through VPN server**

Adapted from *Computer & Internet Security* by Du

VPNs allow secure access to private LAN from outside as if device is inside

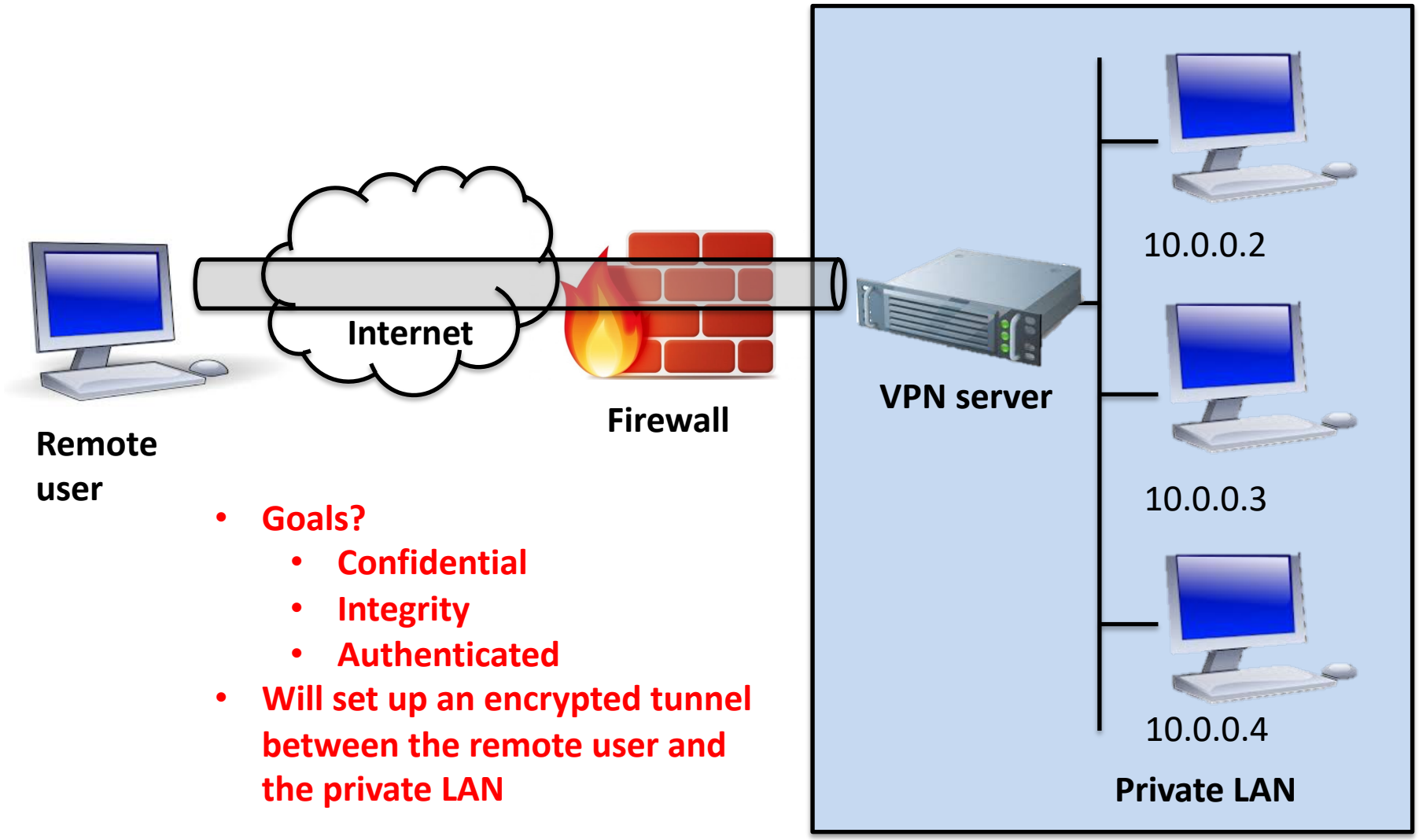


- VPN server exposed to outside
- Can send/receive receive traffic through firewall
- VPN authenticates outsiders, creates encrypted tunnel

All other hosts on inside with non-routable addresses

Adapted from *Computer & Internet Security* by Du

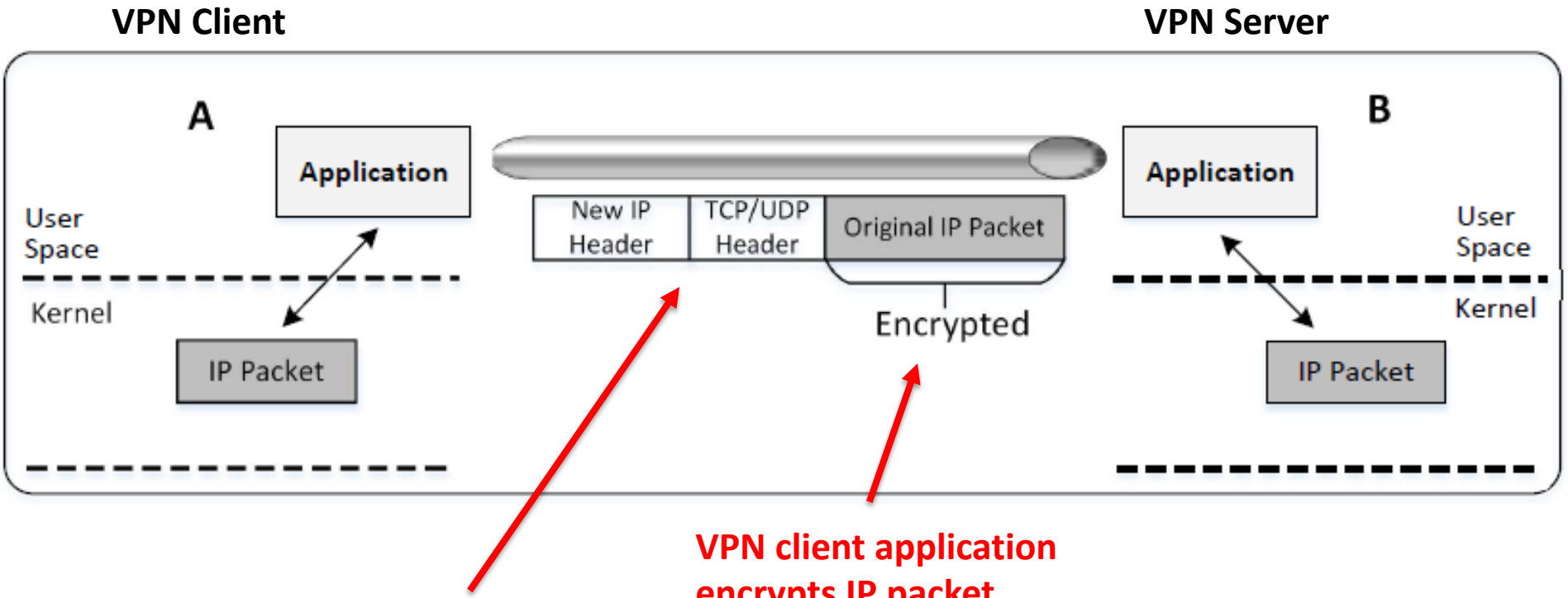
All traffic from outside to inside devices must come through VPN server



- Goals?
  - Confidential
  - Integrity
  - Authenticated
- Will set up an encrypted tunnel between the remote user and the private LAN

Adapted from *Computer & Internet Security* by Du

Transport Layer tunneling securely sends packets to another network in three steps



VPN client application encrypts IP packet

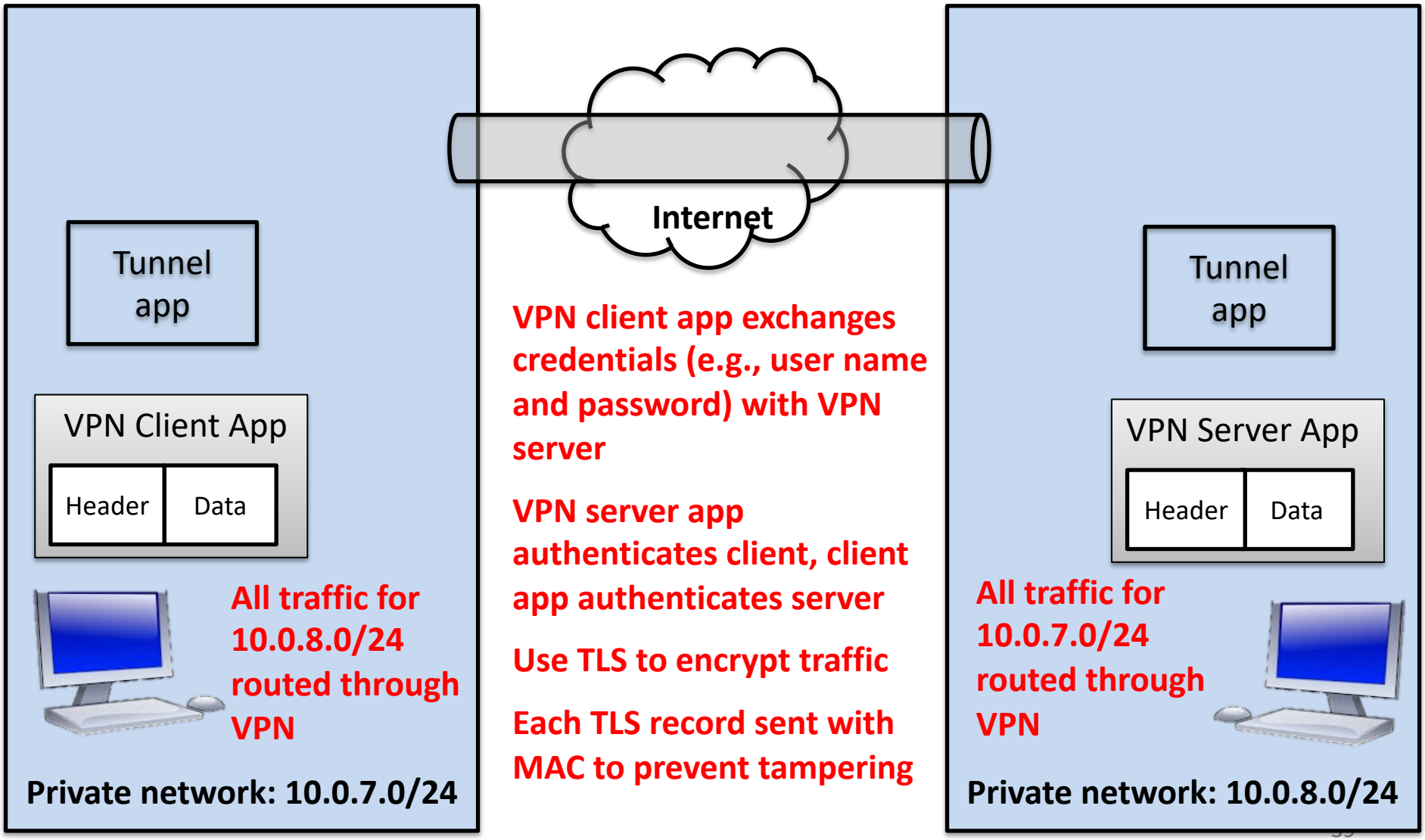
VPN client app adds new headers addressed to *VPN server*

Original source and destination as well as payload are encrypted

Send packet over TLS connection to server
Someone sniffing on Internet sees packet, but does not know ultimate destination or data
TLS without VPN would reveal destination (here just see the VPN server as destination)
This technique is called TLS tunneling

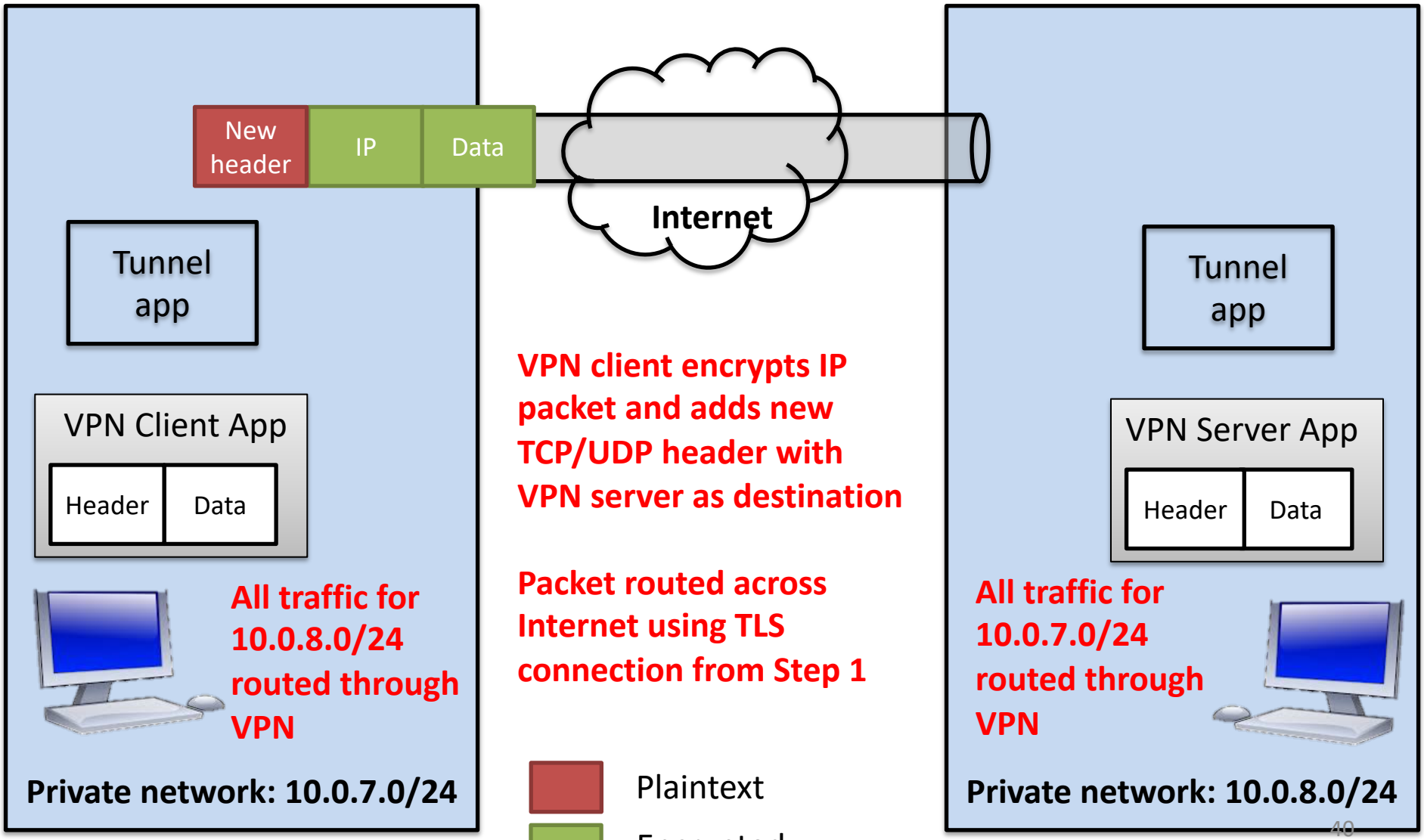
Adapted from *Computer & Internet Security* by Du

Step 1: Establish a TLS tunnel



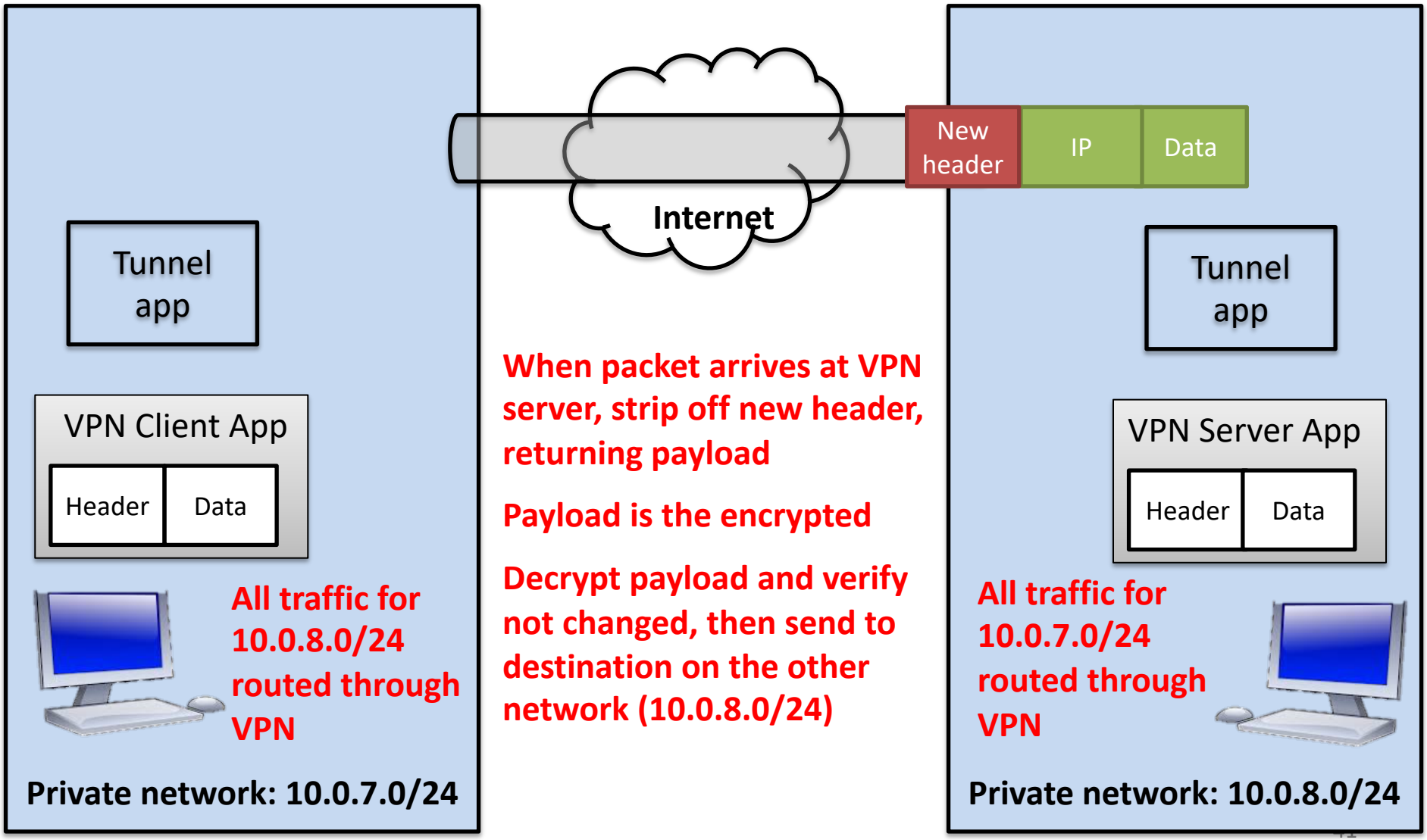
Adapted from *Computer & Internet Security* by Du

Step 2: Forward IP packets



Adapted from *Computer & Internet Security* by Du

Step 3: Release IP packets on the other network



Adapted from *Computer & Internet Security* by Du

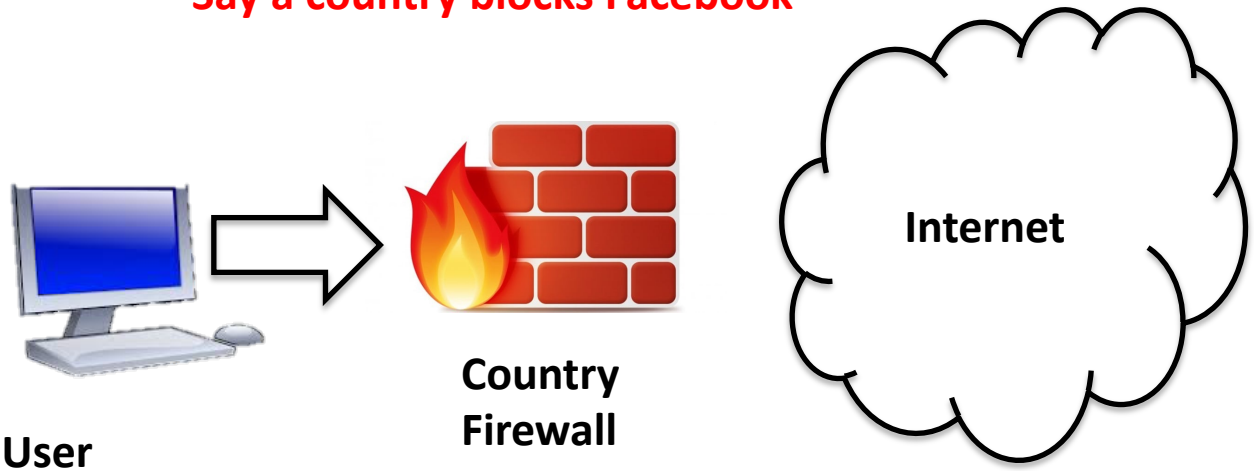
Apart from tunneling, VPNs can be used to bypass firewalls

Image a country has implemented a firewall to prevent access to some international Internet sites

- **Firewall looks at destination IP address**
- **Drops packet if address is on blacklist**

Facebook

Say a country blocks Facebook



User

Country Firewall

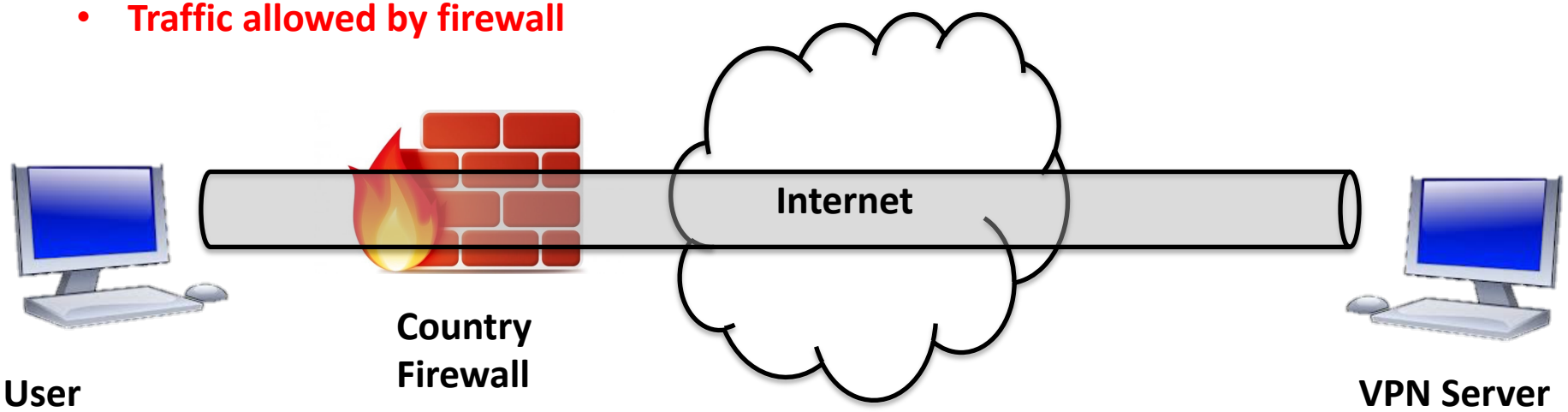
VPN Server

Apart from tunneling, VPNs can be used to bypass firewalls

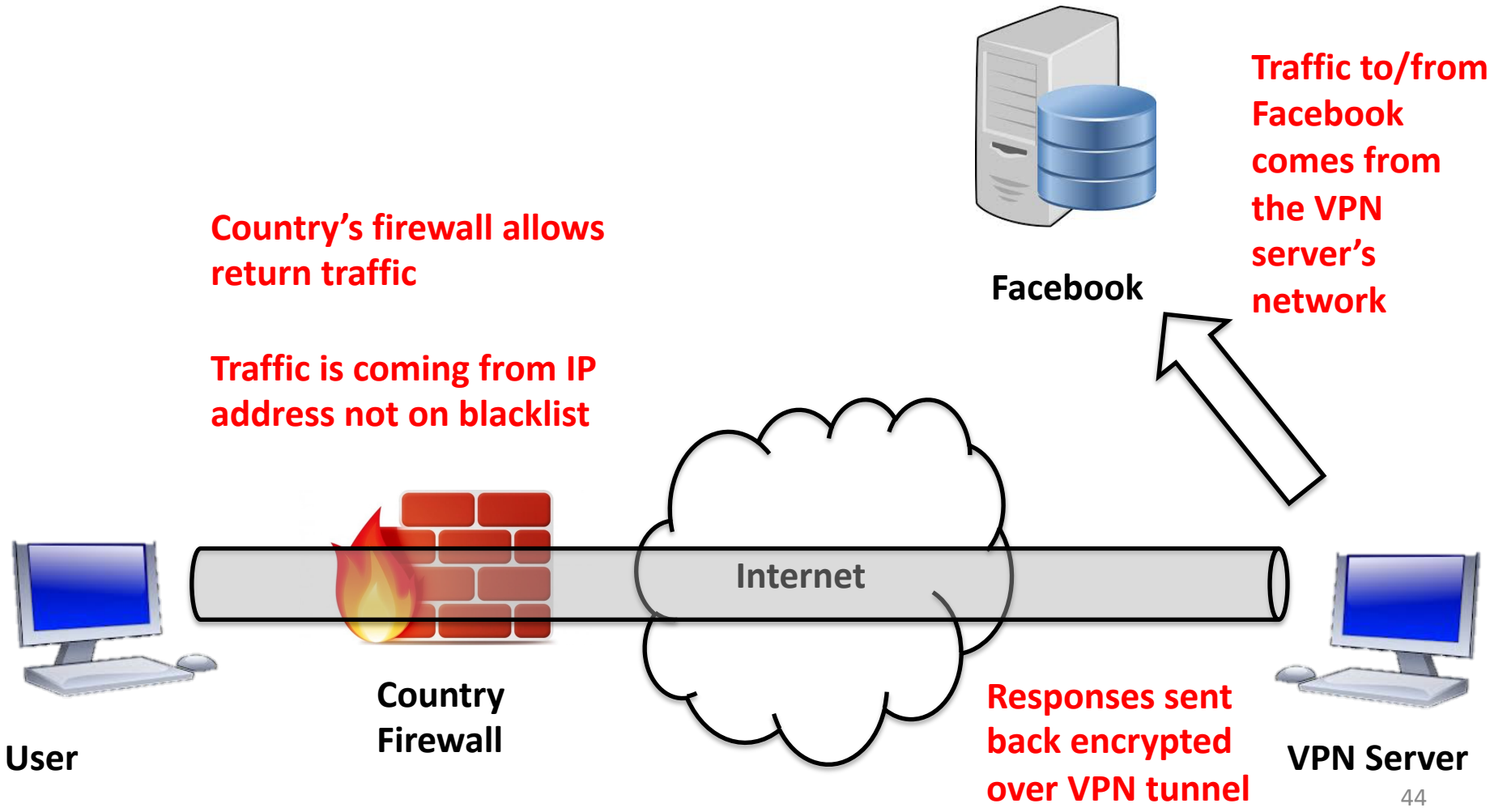
To bypass firewall

- **Establish VPN to server not on blocked list**
- **Say VPN server is in another country and VPN server IP address not block by country firewall**
- **Firewall cannot see real destination (e.g., Facebook) because IP headers encrypted inside VPN tunnel**
- **Firewall only knows packet bound for VPN server**
- **Traffic allowed by firewall**

Facebook



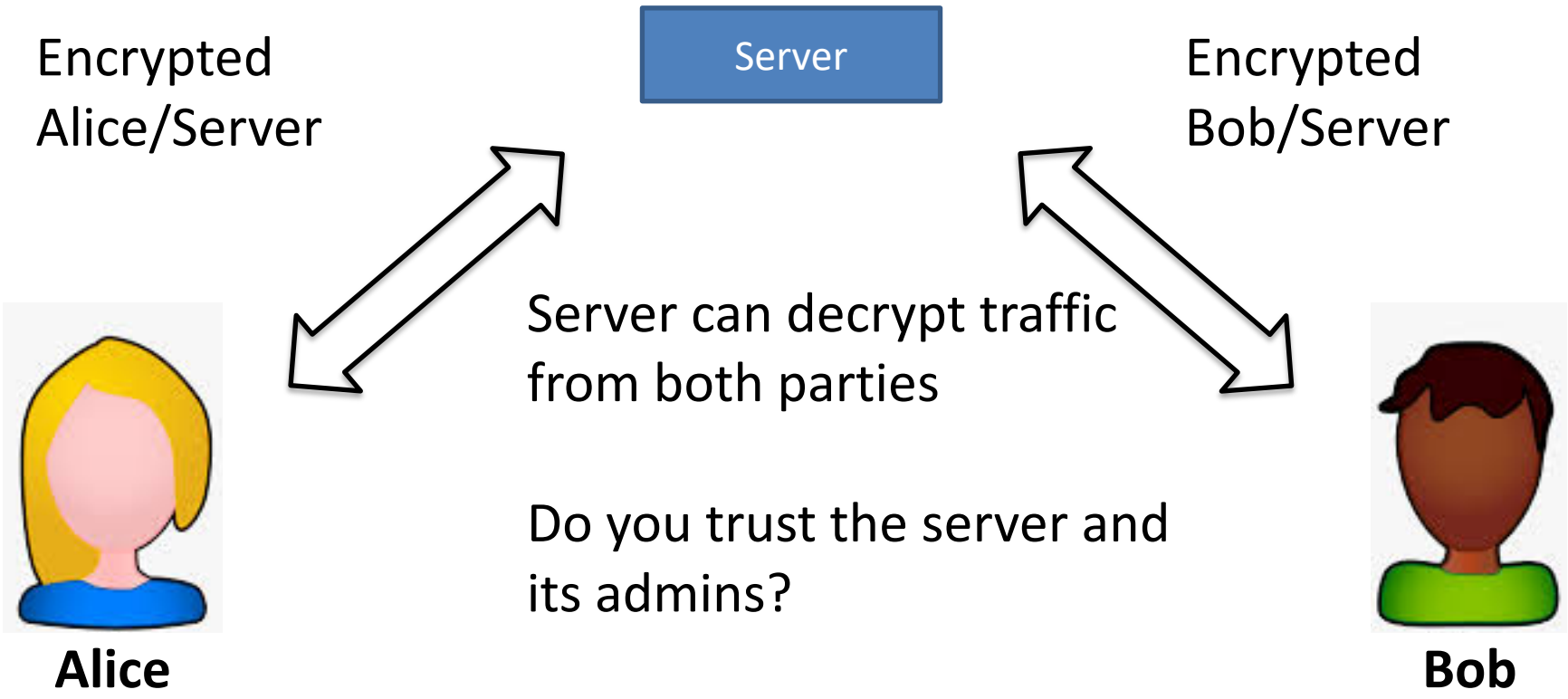
Apart from tunneling, VPNs can be used to bypass firewalls



Agenda

1. The Onion Router (TOR)
2. Transport Layer Security (TLS)
3. Virtual Private Networks (VPNs)
4. Signal/WhatsApp

When sending messages, often an intermediary server can read text



If you don't trust the server, better if things were end-to-end encrypted

Problem: Alice and Bob might not be online at the same time
Also, would like forward and backward secrecy
Could use PGP, but hard to use

Signal is an app that uses built-in end-to-end (E2EE) encryption to send messages

Alice to send message to Bob

Server

Alice gets pre-key bundle with 1 one-time key

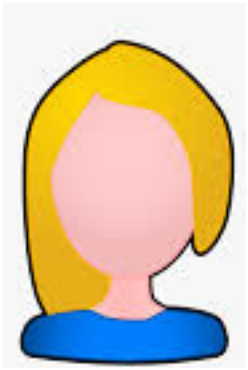
IPK_B
SPK_B
OPK_{Bi}

Works if Alice and Bob not online at same time

Server cannot read

IPK_B
SPK_B
OPK_{B1}
OPK_{B2}
…
OPK_{Bn}

Bob generates Identity key pair, Signed key pair (signed with IPK_B) and *n* one-time key pairs



Alice

IPK_A
EPK_A

Alice has previously generated identity key pair and now generates ephemeral key

What if someone suddenly starts using Signal?

Bob

$KDF\ (DH(IPK_A,SPK_B)\ ||\ DH(EPK_A,IPK_B)||DH(EPK_A,SPK_B)||DH(EPK_A,OPK_B)$

Concatenate results of DH using four different keys
Pass through Key Derivation Function (like hash)

Bob does the same to decrypt

Use this as master key to create "ratchet" function that creates a new key for each message

Signal can visually confirm master keys as QR code (if in same place)

Use RSA with message key, include IPK_A and EPK_A

Adapted from *Security Engineering,* 3rd Edition, by Ross Anderson
https://www.whatsapp.com/security/