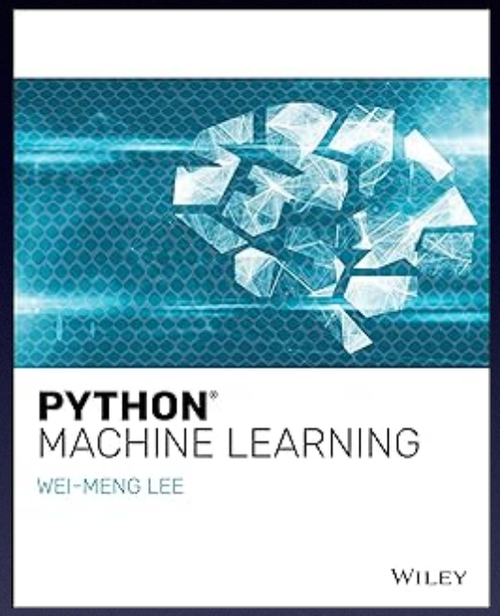
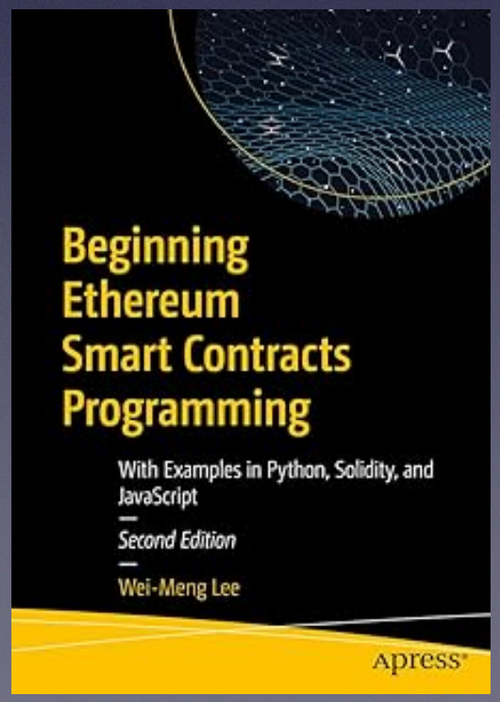
Networking

Wei-Meng Lee
Technologist & Founder
Developer Learning Solutions
http://calendar.learn2develop.net
weimenglee@learn2develop.net

Introduction

- Associate instructor for Neueda
- Developer Learning Solutions (http://calendar.learn2develop.net)
- Software development and training
  - Mobile, Blockchain, Smart Contracts, AI, Large Language Models, Python, Data Analytics, Machine Learning, Deep Learning, etc

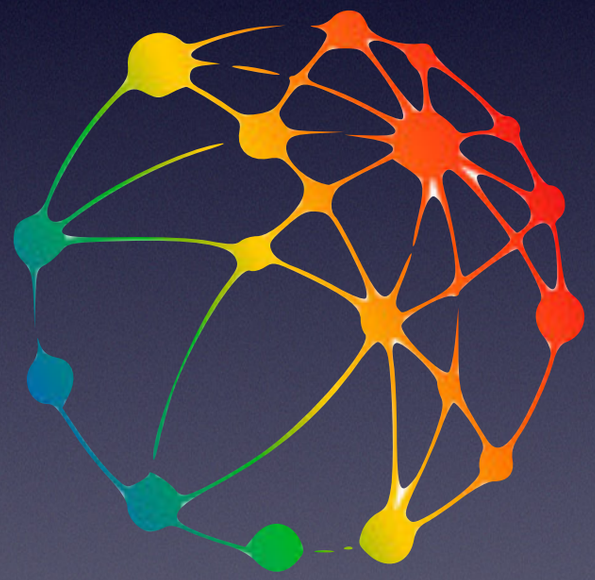




neueda

Developer Learning Solutions

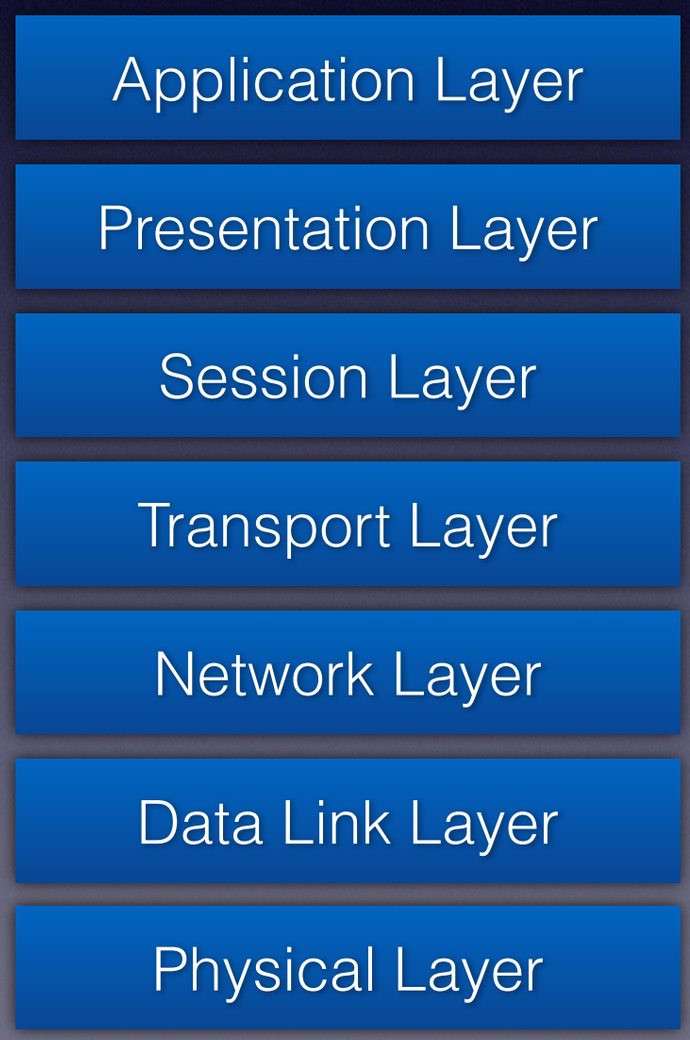
Networking



(c) Wei-Meng Lee weimenglee@learn2develop.net

OSI (Open Systems Interconnection) Layers

- Conceptual framework used to understand and implement network communications between different systems, without regard to their underlying internal structure and technology.



(c) Wei-Meng Lee weimenglee@learn2develop.net

An easy way to understand the OSI layers is to use an analogy of sending a letter to your friends

Penpals - 笔友

OSI Layers

Application Layer	Writing the letter; HTTP, FTP, SMTP, etc
Presentation Layer	Formatting the letter; encryption/decryption, compression, etc
Session Layer	Preparing the envelope; manage connections between apps
Transport Layer	Writing the address; TCP / UDP
Network Layer	Selecting the post office; Internet Protocol (IP)
Data Link Layer	Handling over to the mail carrier; Ethernet and WiFi
Physical Layer	Delivering the letter; physical medium, cables, fibre optics, etc

(c) Wei-Meng Lee weimenglee@learn2develop.net

TCP/IP

- The TCP/IP model, also known as the Internet Protocol Suite, is the conceptual framework used for the Internet and similar networks.
  - Only four layers
  - Each layer corresponds to one or more layers of the OSI model

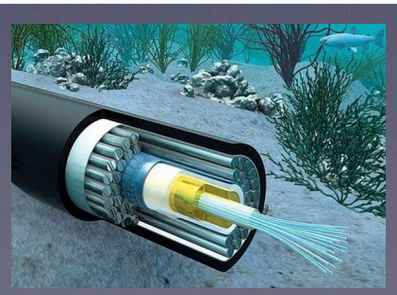
Layered Approach to Networking

English 中文
español

Application HTTP, FTP, SMTP

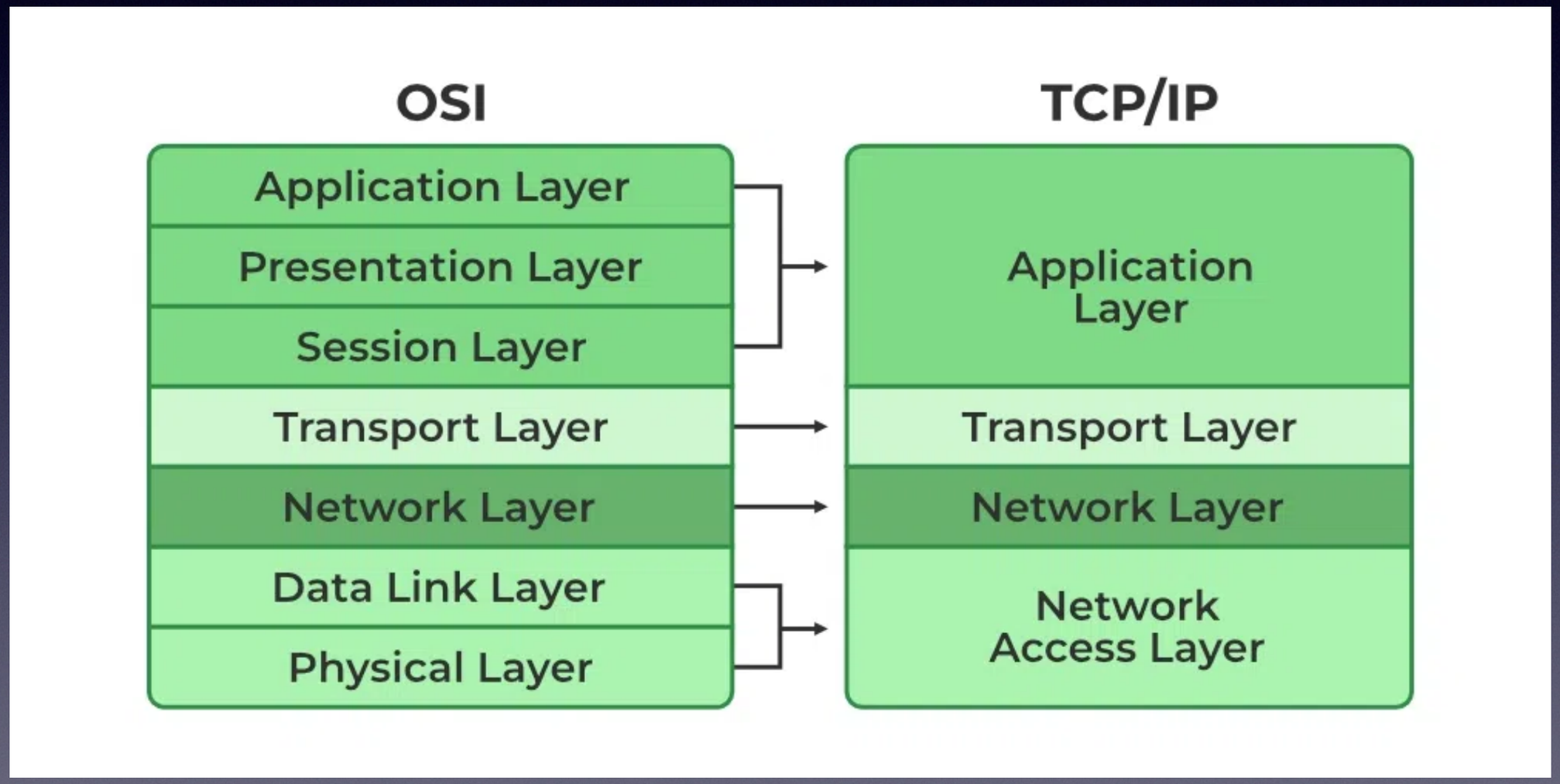
Transport TCP, UDP

Internet IP, ICMP, ARP



Network Interface Ethernet, WiFI

OSI vs TCP/IP


OSI
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
TCP/IP
Application Layer
Transport Layer
Network Layer
Network Access Layer

Layers of TCP/IP

- Application Layer
- Transport Layer
- Internet Layer
- Network Access (Data Link) Layer

Application Layer

- This layer is where most people are familiar with
  - responsible for the communication between end-user applications and the lower layers of the TCP/IP stack
- Examples:
  - HTTP/HTTPS
  - FTP
  - SMTP / POP / IMAP
  - SSH

Transport Layer

- This layer ensures data transmission between two devices is reliable.
  - Breaks down large messages into smaller packets, sends those packets to the target machine, and then the target machine reassembles the packets back into the original message.
- Examples
  - **TCP** - A connection-oriented protocol that ensures data sent from one computer to another arrives intact and in the correct order
    - Think of this as making a phone call - what Person A says arrives in the same order as is heard by Person B
  - **UDP** - A connectionless protocol often used for streaming services. It's faster than TCP but doesn't guarantee the order of data packets.
    - Think of this as sending two letters - the order the letters are received may not be in the same order as they were sent

Questions

- Name me some apps that use TCP
- Name me some apps that use UDP

Answers

- Name me some apps that use TCP
  - Web browsers
  - Emails protocols such as SMTP, IMAP, and POP3
  - FTP, SSH
  - Databases
  - WhatsApp - messaging
- Name me some apps that use UDP
  - Skype, Zoom, WhatsApp - for real-time video and voice communications
  - Online games like Fortnite, Call of Duty, and Legend of Legends
  - Spotify
  - DNS queries

Internet (Network) Layer

- The post office of the internet. It's responsible for addressing, packaging, and routing data to its destination.
- Examples:
  - **IP** - Assigns addresses to devices and routes data based on these addresses. Your device's IP address is its unique identifier on the internet.
  - **ICMP** - Used for error handling and diagnostics. If you've ever used the "ping" command, you've used ICMP.

Network Access (Data Link) Layer

- Deals with the physical connection between devices. Whether it's through a wired Ethernet connection or a wireless Wi-Fi signal, the link layer is all about getting data on and off the physical network medium.

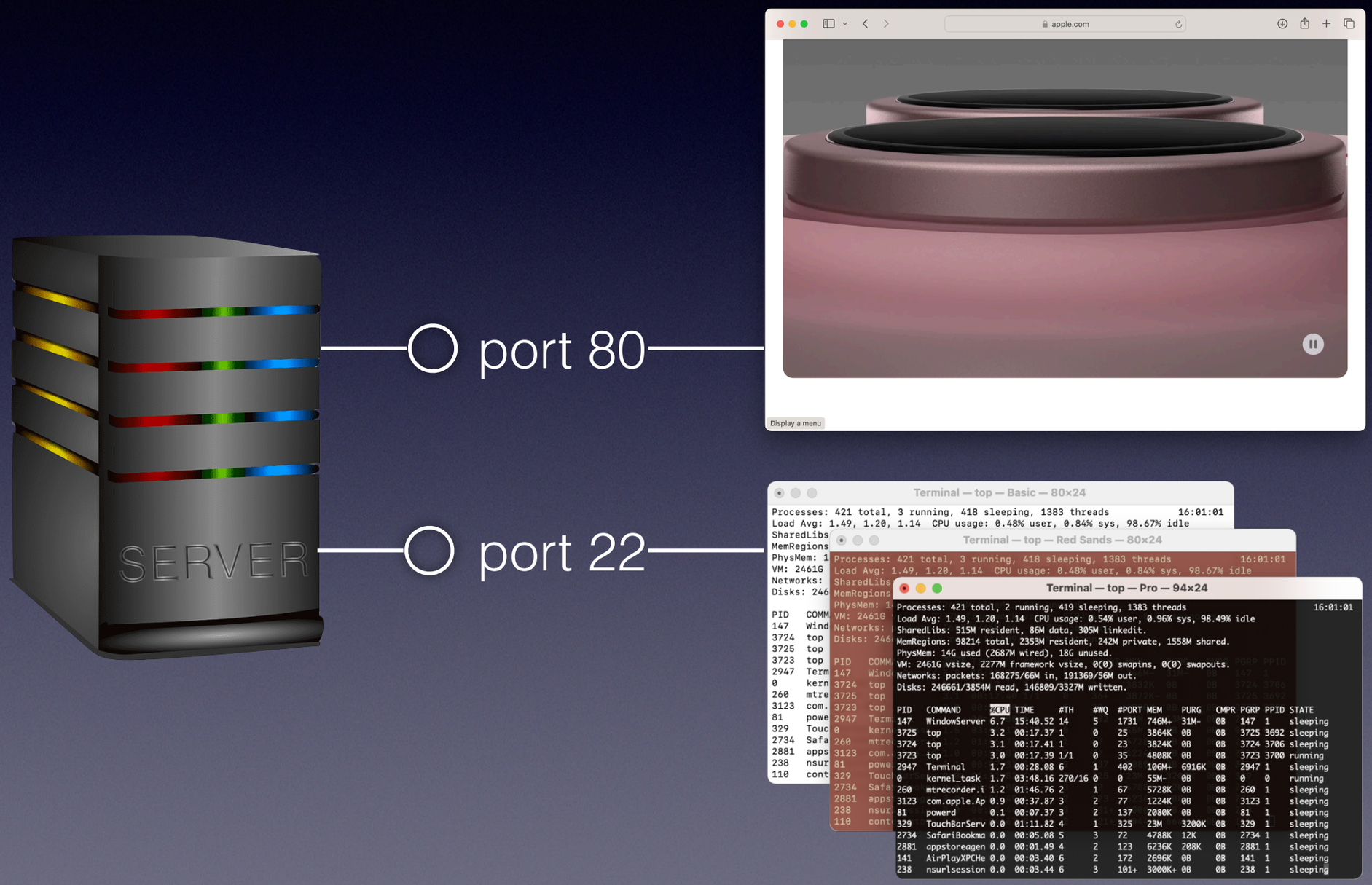
- Examples:
  - **Ethernet** - The common wired networking technology
  - **Wi-Fi** - The protocol for wireless networking.
  - **ARP** - Finds the hardware address of a host from a known IP address, ensuring data gets to the right device on a local network.

What type of cable is this?

Port Numbers

- A port is a logical channel through which data is sent and received
- Each port is associated with a service or application on a device
- Ports range from 0 to 65,535
  - Ports 0 to 1023 are reserved for well-known applications
  - E.g. http://www.apple.com:80

Ports on a computer



(c) Wei-Meng Lee weimenglee@learn2develop.net

Quiz

- Give me the service name for the following ports:
  - 20,21
    - FTP
  - 23
    - Telnet
  - 25
    - SMTP
  - 143
    - IMAP

Quiz

- Give me the service name for the following ports:
  - 443
    - HTTPS
  - 3306
    - MySQL
  - 1433
    - Microsoft SQL Server
  - 1521
    - Oracle

Understanding IP addresses

(c) Wei-Meng Lee weimenglee@learn2develop.net

A Revision of Binary System

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	0	0	0	0	1	1	1
Binary								
Decimal	128		+			4 + 2 + 1 = 135		

(c) Wei-Meng Lee weimenglee@learn2develop.net

Range of values representable

- For a 8-bit *unsigned* number, range is 0 to 2^8-1, or **0 to 255**
- For a 8-bit *signed* number, range is -2($^{8-1)}$ to 2($^{8-1)}$-1, or **-128 to 127**

Hexadecimal

- Valid characters are from 0 to 9, and A to F
- Each character takes up 4 bits

1	1	1	0	0	1	1	1	Binary
14				7				Decimal
E				7				Hexadecimal

IPv4

123.88.45.75

8-bit	8-bit	8-bit	8-bit

32-bit = 4 bytes

IPv6

Hexadecimal

`2001:0db8:85a3:0000:0000:8a2e:0370:7334`

16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit

128-bit = 16 bytes

Can be shortened to:

`2001:db8:85a3::8a2e:370:7334`

IP Address Classes

- **Class A**: IP addresses between 1.0.0.0 to 126.0.0.0
- **Class B**: IP addresses between 128.0.0.0 to 191.255.0.0
- **Class C**: IP addresses between 192.0.0.0 to 223.255.255.0
- **Class D** or multicast: IP addresses between 224.0.0.0 to 239.255.255.255
- **Class E**, which are reserved for experimental usage: IP addresses between 240.0.0.0 to 255.255.255.255

(c) Wei-Meng Lee weimenglee@learn2develop.net

Home Networks

- For home networks, the most commonly used IP address class is **Class C**.
  - Range: Class C addresses range from **192.0.0.0 to 223.255.255.255**.
  - Subnet Mask: The default subnet mask for Class C addresses is 255.255.255.0, which provides enough addresses for up to 254 devices per network (with some addresses reserved for network identification and broadcasting).
  - Suitability: Class C addresses are suitable for small networks, which is typically the case for home networks with a limited number of devices.
- These addresses are not routable on the public internet, providing a level of security and privacy for internal network communications within the home.

Subnet Mask

(c) Wei-Meng Lee weimenglee@learn2develop.net

Subnet Mask

- Splits IP address into host and network addresses
  - A **subnet mask** is a 32-bit number created by setting host bits to all 0s and setting network bits to all 1s. In this way, the subnet mask separates the IP address into the network and host addresses.

IP address: `192.168.1.100`

Subnet mask: `255.255.255.0`

Also known as 192.168.1.100/24 in CIDR notation

IP address: `11000000.10101000.00000001.01100100`

Subnet mask: `11111111.11111111.11111111.00000000`

Network Host

- Performing an **AND** on the IP address and the Subnet mask gives you the *network* address
- To find the *host* address, invert the subnet mask and then perform an **AND** with the IP address

Subnet Mask

- Splits IP address into host and network addresses
  - A **subnet mask** is a 32-bit number created by setting host bits to all 0s and setting network bits to all 1s. In this way, the subnet mask separates the IP address into the network and host addresses.

IP address: `172.31.26.190`

Subnet mask: `255.255.240.0`

Also known as 172.31.26.190/20 in CIDR notation

IP address: `10101100.00011111.00011010.10111110`

Subnet mask: `11111111.11111111.11110000.00000000`

Network | Host

- Network address is 172.31.16.0
- Host address is 0.0.10.190

Question

- Given a machine IP address of 172.31.26.190 with subnet mask of 255.255.240.0:
  - Can another computer with an IP address of 172.31.18.190 communicate with the machine above (without using a router)?

Computer 1

172.31.26.190

IP address: 10101100.00011111.00011010.10111110

Subnet mask: 11111111.11111111.11110000.00000000

Network | Host

- Network address is 172.31.16.0

Computer 2

172.31.18.190

IP address: 10101100.00011111.00010010.10111110

Subnet mask: 11111111.11111111.11110000.00000000

- Network address is 172.31.16.0

IP Classes and their users

Class	Number of networks	Number of hosts per network	For who
A	128	16,000,000	Early adopters of the Internet such as large companies and universities.
B	16,384	65,534	Regional Internet Service Providers (ISPs), large educational institutions, and businesses with multiple offices.
C	2,097,152	254	Small businesses, small ISPs, and smaller branch offices.

TATOOINE
DO OR DO NOT...THERE IS TRY.
comic.browserling.com
127.0.0.1 SWEET 127.0.0.1.

(c) Wei-Meng Lee weimenglee@learn2develop.net

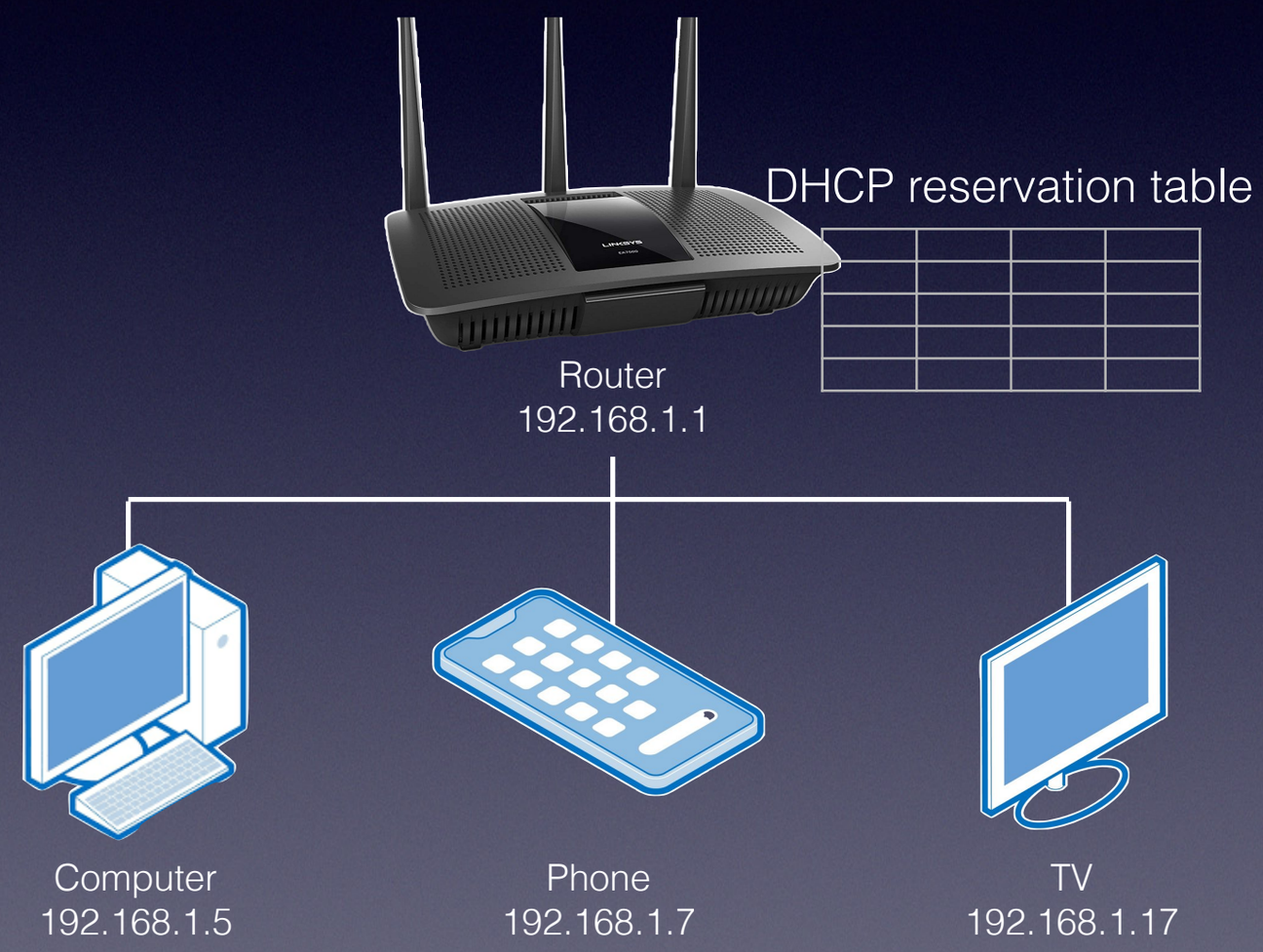
Special IP Addresses

- **0.0.0.0**
  - This IP address in IPv4 is also known as the default network. It is the non-routeable meta address that designates an invalid, non-applicable, or unknown network target.
- **127.0.0.1**
  - This IP address is known as the loopback address, which a computer uses to identify itself regardless of whether it has been assigned an IP address.
- **169.254.0.1 to 169.254.254.254**
  - A range of addresses that are automatically assigned if a computer is unsuccessful in an attempt to receive an address from the DHCP.
- **255.255.255.255**
  - An address dedicated to messages that need to be sent to every computer on a network or broadcasted across a network.

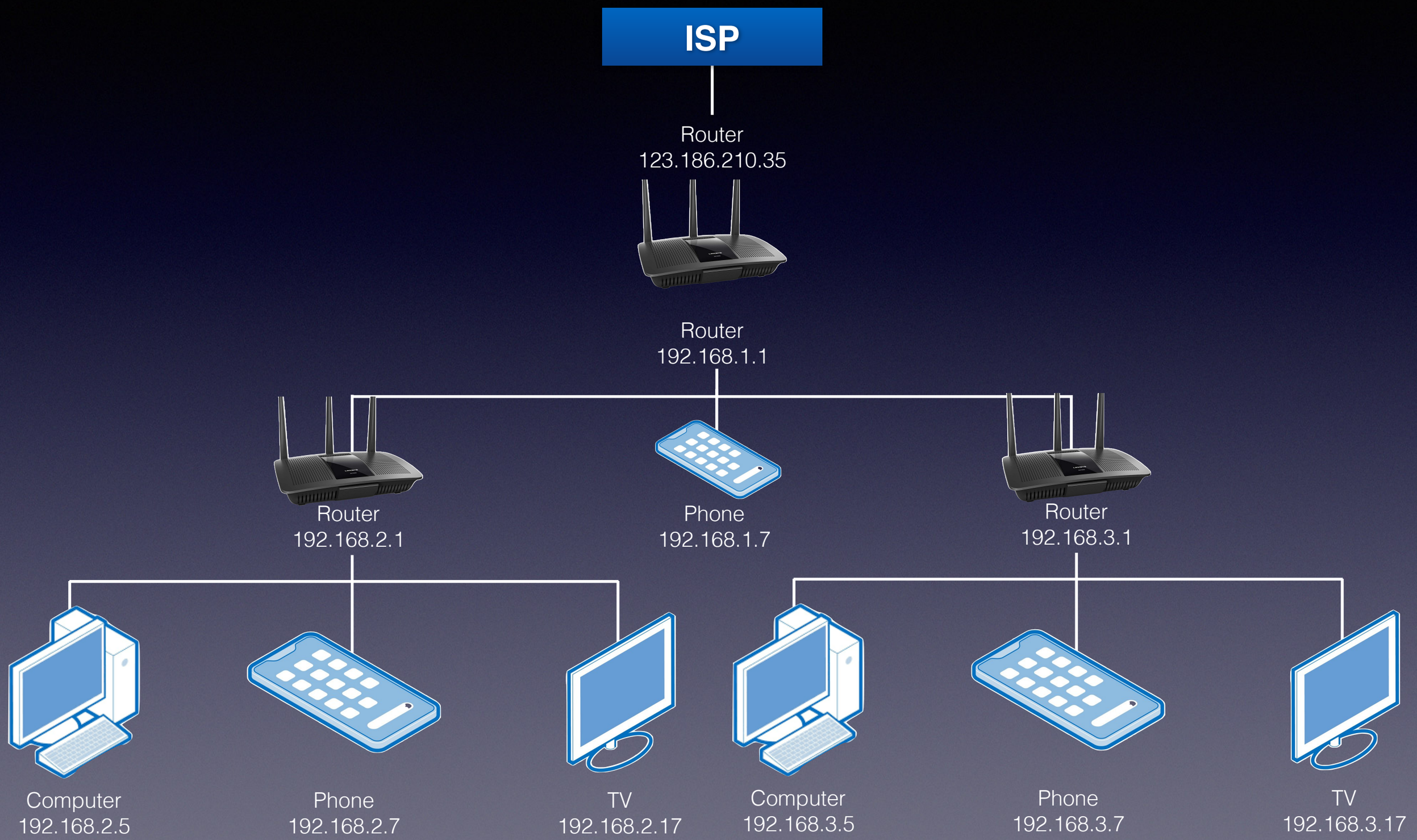
DHCP

DHCP (Dynamic Host Configuration Protocol)

- Router dynamically assigns a new IP address to a device
- The IP address is associated with the MAC address of the network interface of the device
- You can configure the router to reserve a fixed IP address for a device based on its MAC address
- Usually when you reconnect to the router before its lease expires, you will get back the same IP address, unless the IP address has been assigned to some other devices



Multiple Subnets



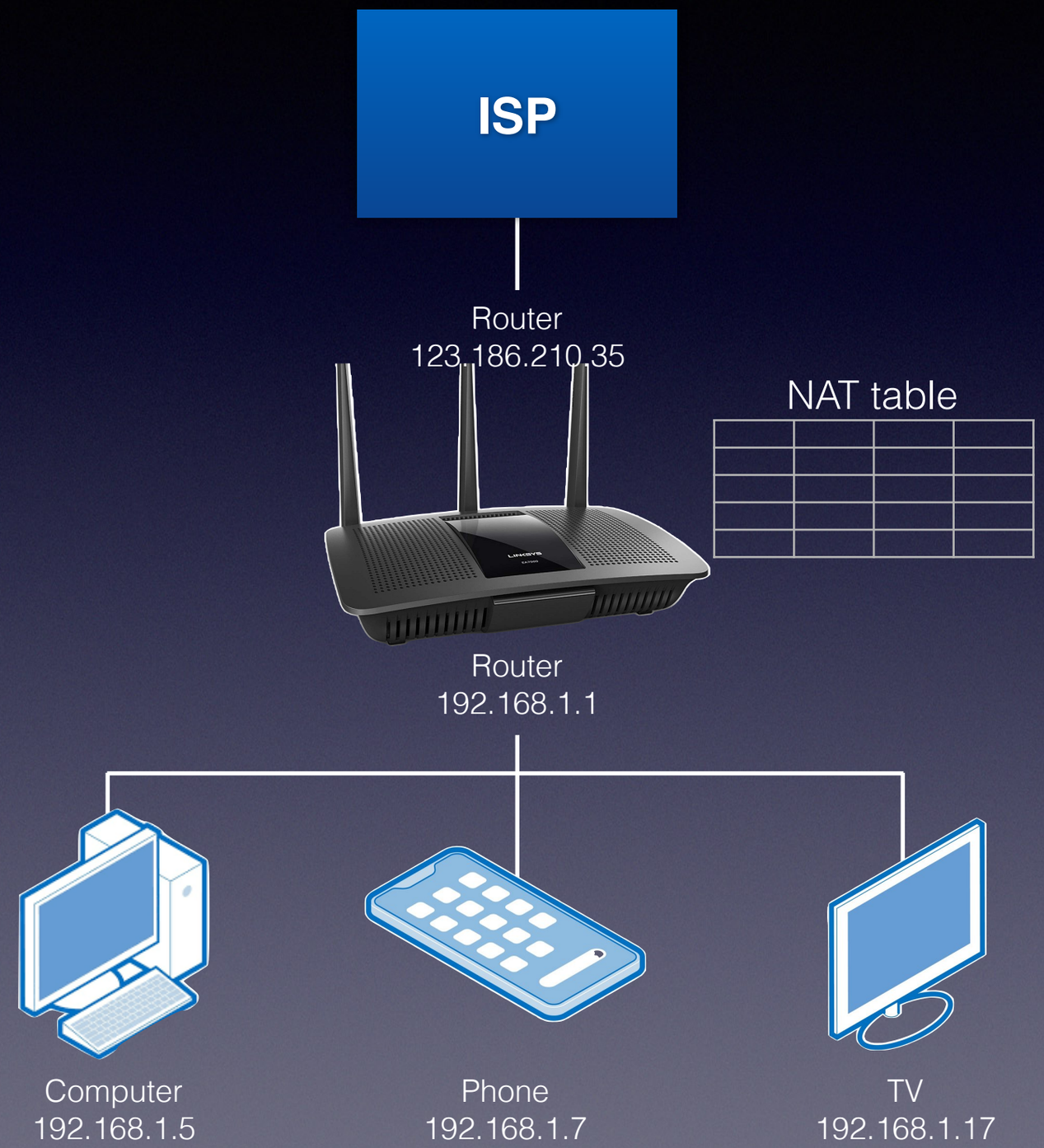
(c) Wei-Meng Lee weimenglee@learn2develop.net

NAT

(c) Wei-Meng Lee weimenglee@learn2develop.net

Network Address Translation (NAT)

- The router maintains a NAT (Network Address Translation) table
- For e.g. a device with IP address 192.168.1.5 sends a request to a web server
- The router translates it to the public address of the router (203.0.113.5) and an assigned port (e.g. 15000)
- When the web server responds, the router translates the public IP address and port back to the original device's IP address and the original private port



MAC Address

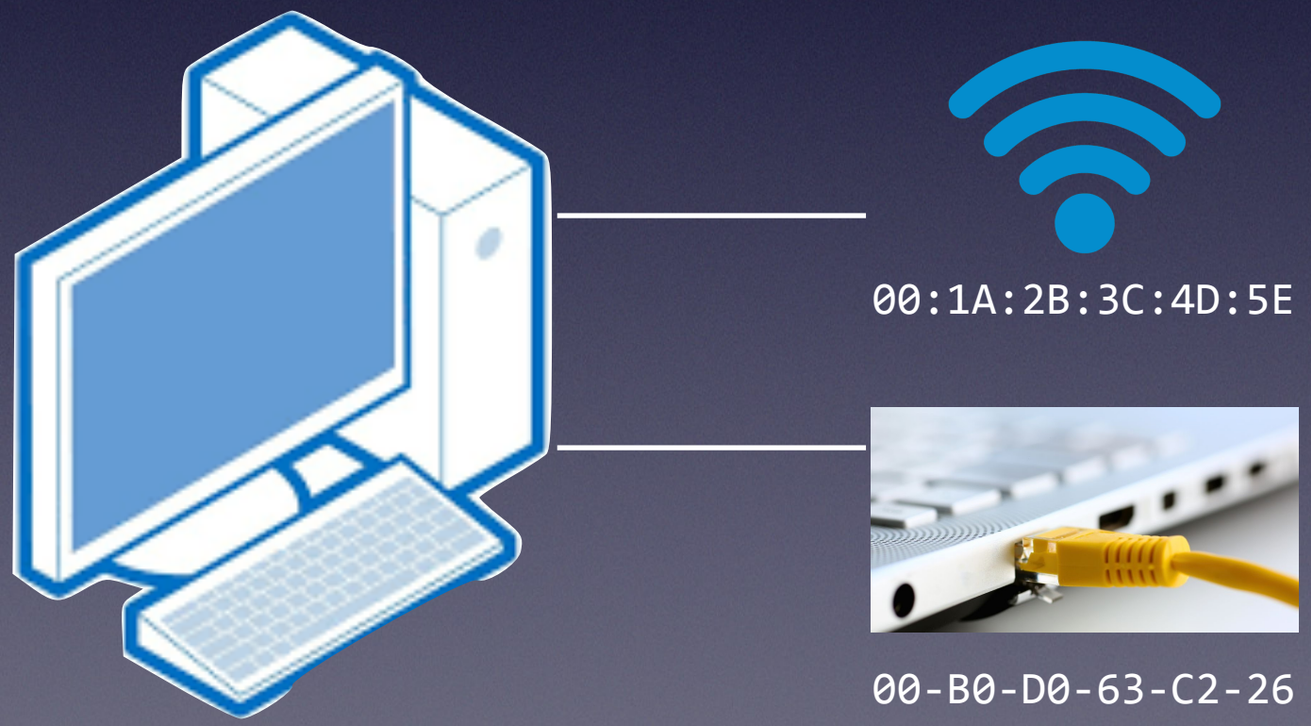
(c) Wei-Meng Lee weimenglee@learn2develop.net

MAC Address

- Media Access Control - MAC
  - Every network device has a unique MAC address assigned to its network interface card (NIC). The MAC address is used to identify the device on the local network.

00	1A	2B	3C	4D	5E
8-bit	8-bit	8-bit	8-bit	8-bit	8-bit

48-bit = 6 bytes

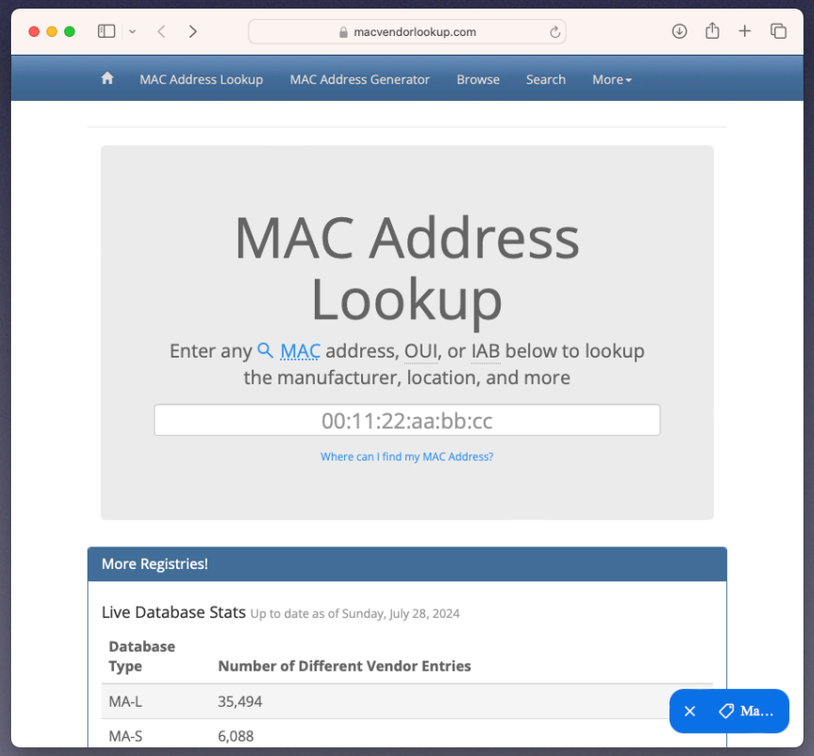


MAC Address

Manufacturer			Device ID		
00	1A	2B	3C	4D	5E
8-bit	8-bit	8-bit	8-bit	8-bit	8-bit

48-bit = 6 bytes

- Apple, Inc.: `00:03:93`
- Cisco Systems, Inc.: `00:14:6C`
- Intel Corporation: `00:1B:21`
- Microsoft Corporation: `00:23:7D`
- Samsung Electronics Co., Ltd.: `00:11:22`

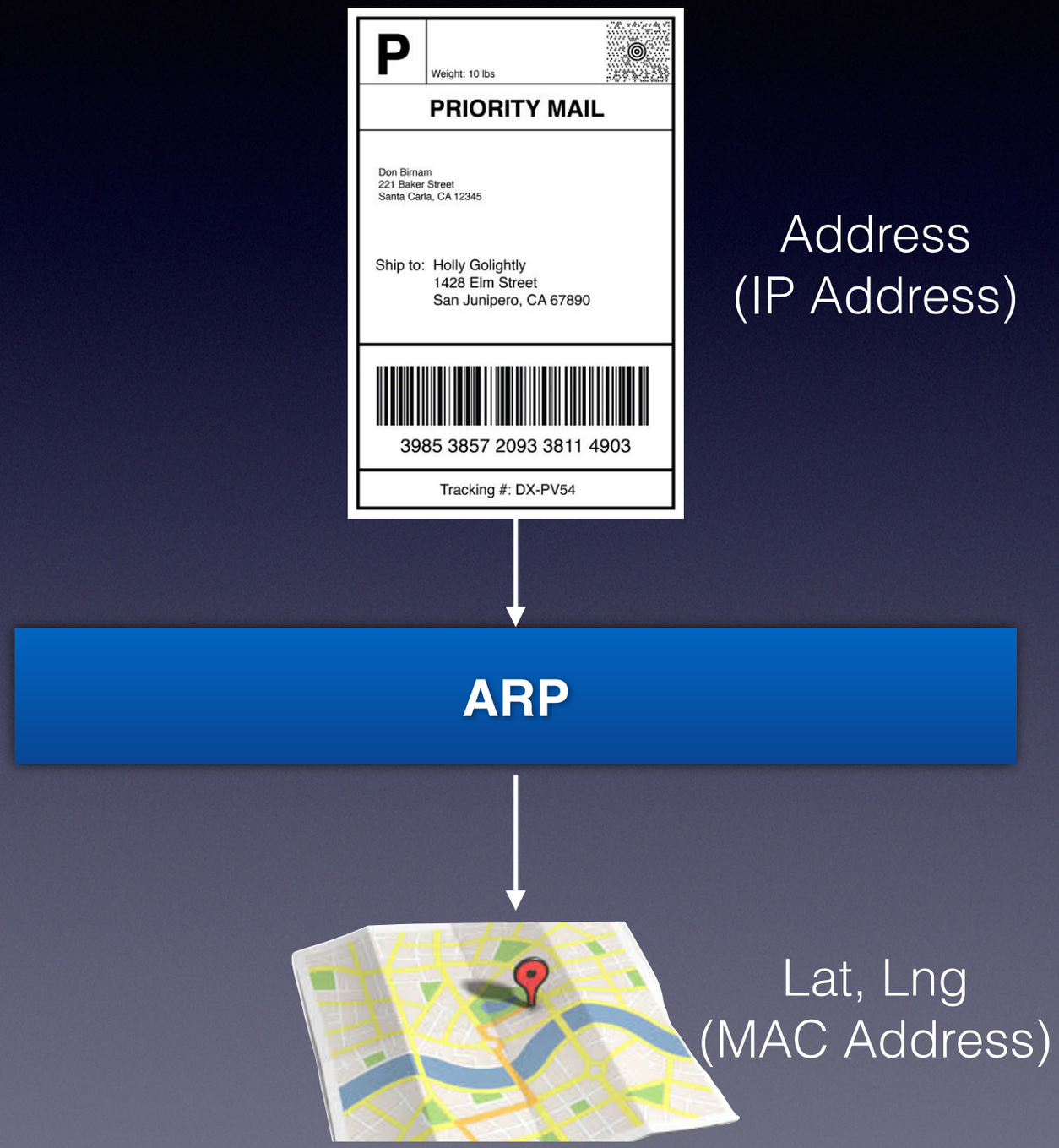


ARP

(c) Wei-Meng Lee weimenglee@learn2develop.net

ARP

- ARP (Address Resolution Protocol) is used to map or translate IP addresses to physical MAC (Media Access Control) addresses within a local area network (LAN)
- Think of ARP as the delivery personnel who converts your package address to actual location (Lat and Long)
- There is also Reverse ARP (RARP) - converts from MAC address back to IP address
  - RARP is obsolete and replaced by DHCP



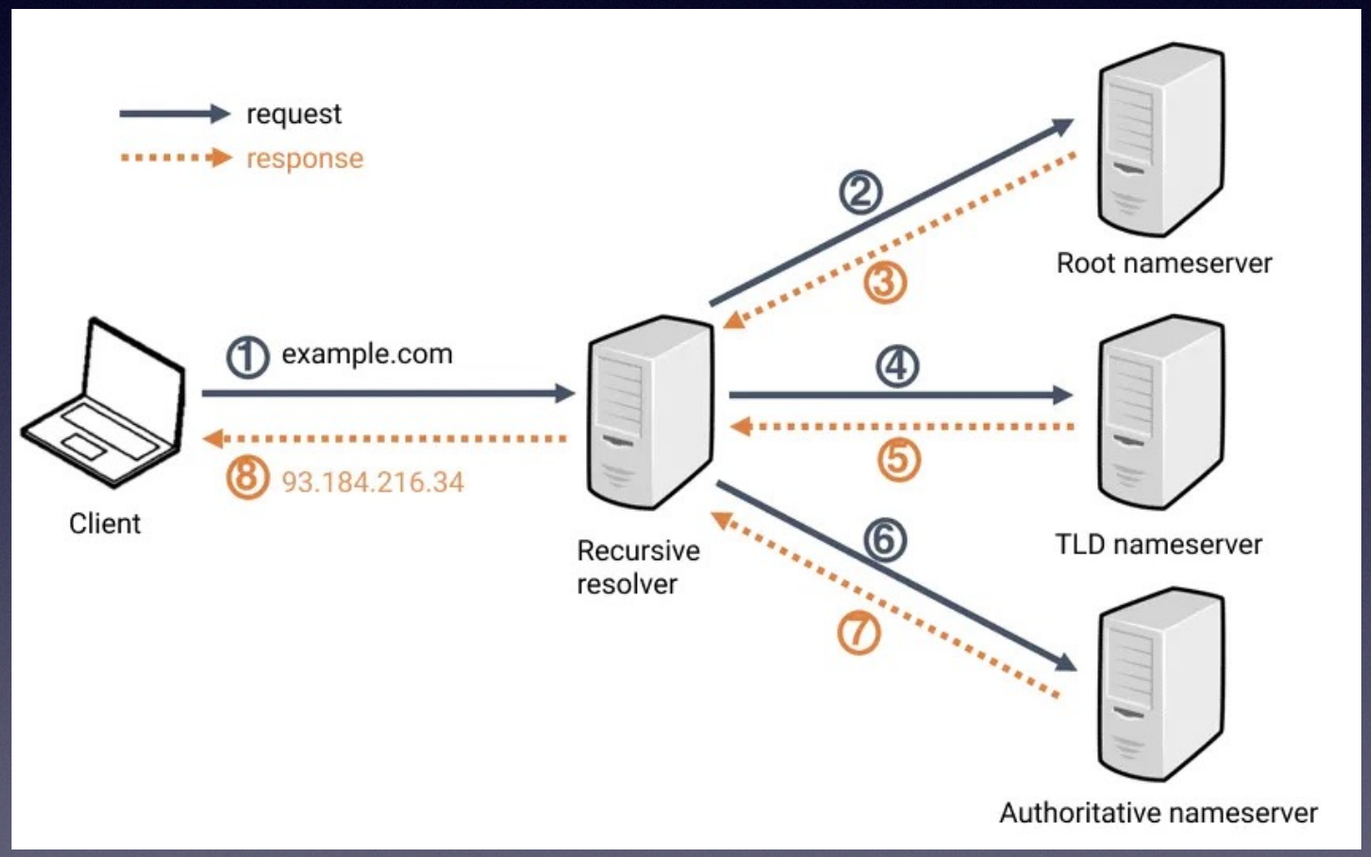
Which Devices using ARP?

- **Computers and Laptops:**
  - These devices use ARP to resolve IP addresses to MAC addresses on a local network to send and receive data packets.
- **Network Routers:**
  - Routers use ARP to determine the MAC address of devices within the same local network. This helps in routing packets correctly within the local network.
- **Switches:**
  - Network switches use ARP to populate their MAC address tables, which are used to forward packets to the correct port.
- **Network Printers:**
  - Printers on a local network use ARP to communicate with other devices and accept print jobs.
- **Smartphones and Tablets:**
  - These mobile devices use ARP to interact with other devices on a Wi-Fi network.

Domain Name System (DNS)

- A fundamental component of the internet that translates human-readable domain names (like www.example.com) into machine-readable IP addresses (like 93.184.216.34).

DNS Resolution (1 of 2)



https://medium.com/@soulaimaneyh/discover-what-behind-typing-google-com-into-your-browser-and-pressing-enter-detailed-60bf2679470b

0. **Request Initiation**

- When you enter a domain name in your web browser, your computer sends a DNS resolution request to a DNS resolver. The resolver can be your Internet Service Provider's (ISP) DNS server or a third-party DNS resolver like Google DNS or OpenDNS.

1. Recursive Query

- The DNS resolver receives the request and starts the resolution process. If the resolver doesn't already have the IP address corresponding to the domain name in its cache, it begins a recursive query.

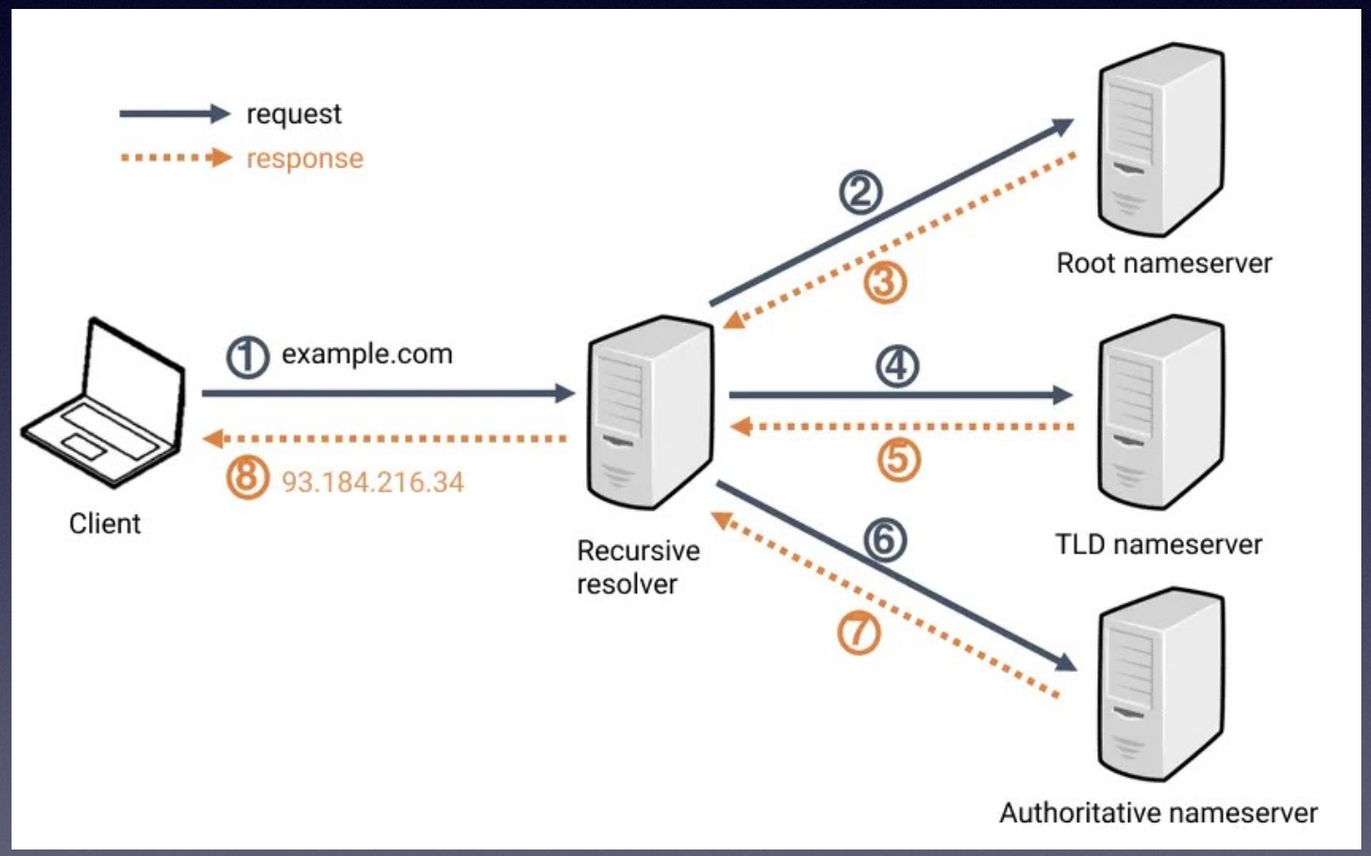
2. Root DNS Servers

- The resolver first contacts the root DNS servers. These servers are responsible for directing queries to the appropriate top-level domain (TLD) servers. There are a relatively small number of root DNS servers distributed worldwide.

3. TLD DNS Servers

- The root DNS servers respond to the resolver with the IP address of the TLD DNS server responsible for the specific top-level domain (e.g., .com, .org, .net) of the requested domain name.

DNS Resolution (2 of 2)



https://medium.com/@soulaimaneyh/discover-what-behind-typing-google-com-into-your-browser-and-pressing-enter-detailed-60bf2679470b

4 and 5. TLD DNS Servers

The resolver then contacts the TLD DNS server and requests the IP address of the authoritative DNS server for the domain name. The authoritative DNS servers are responsible for storing and providing the DNS records for specific domain names.

6. DNS Records

The authoritative DNS server receives the query and checks its records for the requested domain name. If the IP address is found, it is sent back to the resolver.

7. Caching

The resolver receives the IP address from the authoritative DNS server and stores it in its cache for future reference. This caching helps to speed up subsequent DNS resolution requests for the same domain name.

8. Response to Requester

The resolver sends the IP address back to your computer, which can then establish a connection with the website's server using the provided IP address

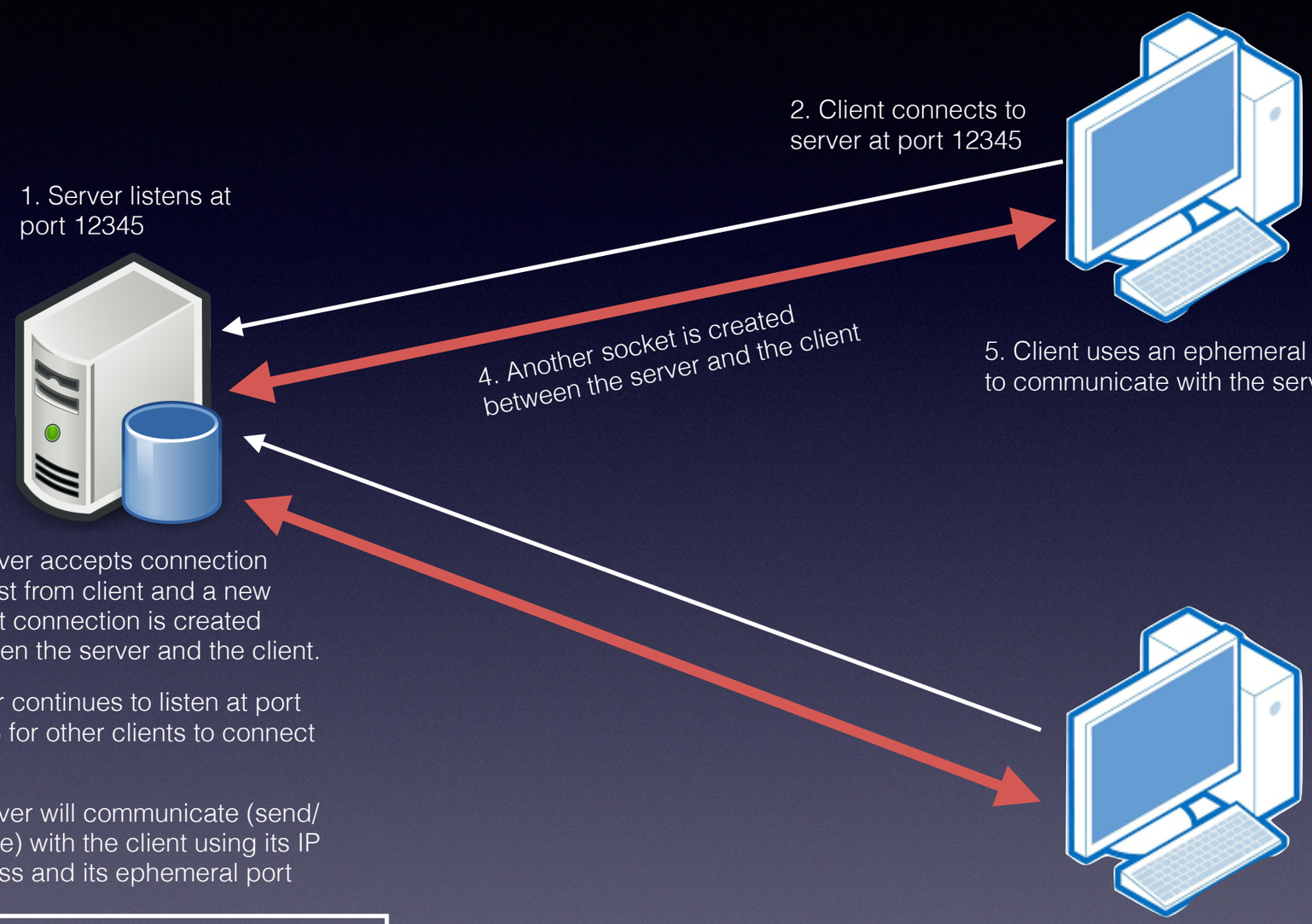
Some Common DNS Servers

- **Google Public DNS**
  - 8.8.8.8
- **Cloudflare DNS**
  - 1.1.1.1
- **OpenDNS**
  - 208.67.222.222

HTTP

- HTTP is a stateless protocol
- Client does not maintain a persistent connection with the server
  - This makes it difficult for the server to send ad-hoc data to the client. E.g. a cinema ticket booking system cannot directly update all clients about the seats' availability
  - You need a way to have a persistent connection between the clients and the server
- The solution is **Sockets / Web Sockets**.

Sockets Programming



Internally, the server maintains a mapping of the client's IP address and port to the specific connection.

Sockets in Your Pockets

Apple Push Notification Service (APNS)

Firebase Cloud Messaging (FCM)

Huawei Push Kit

How HTTPS works

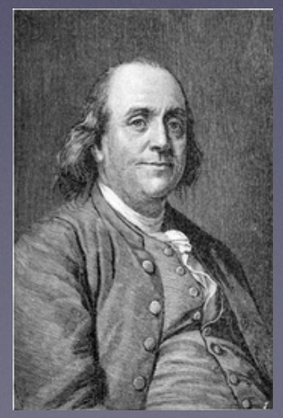
Types of Cryptographic Algorithms

- Types of cryptographic algorithms:
  - Symmetric
  - Asymmetric
  - Hashing

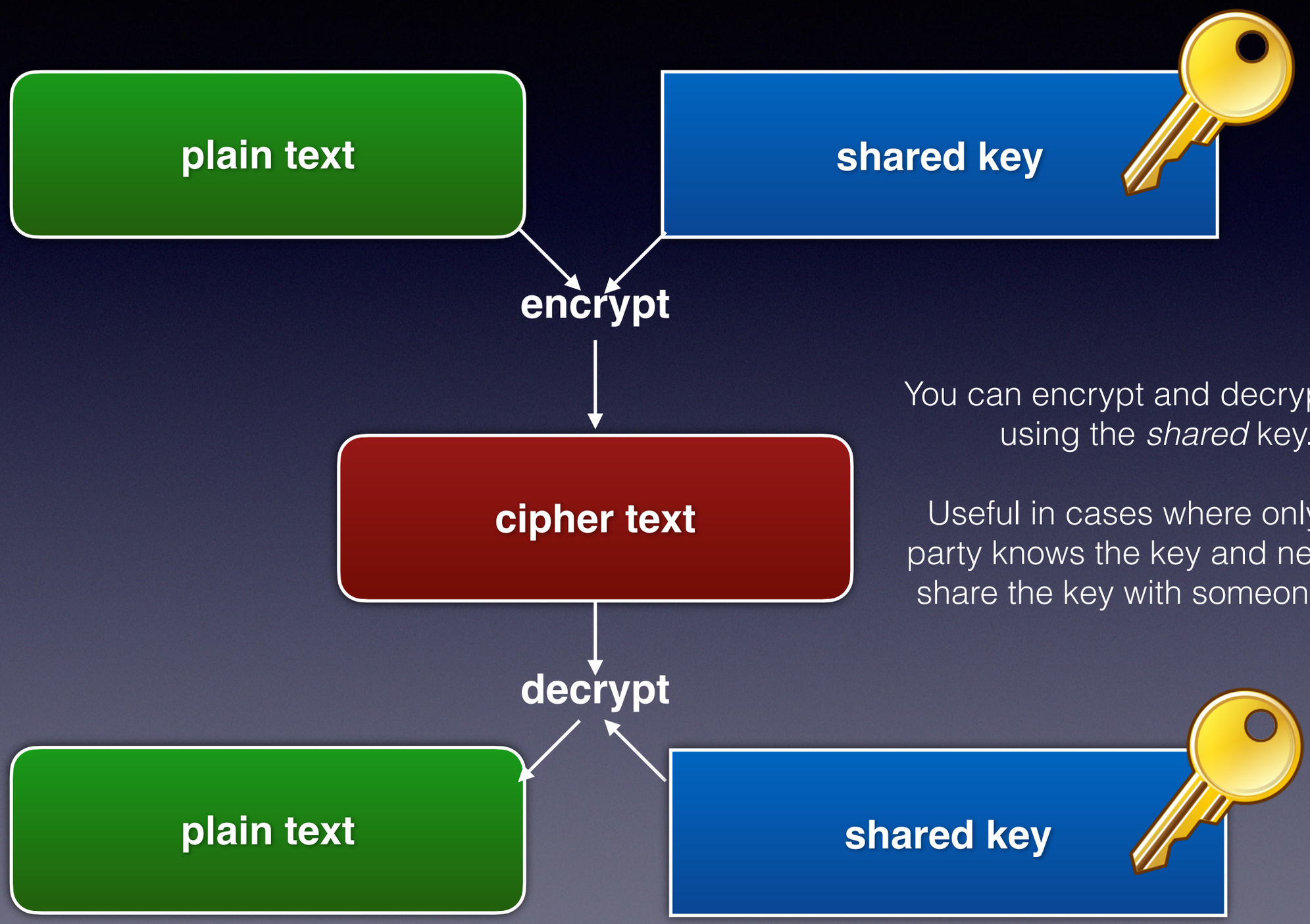
Symmetric Encryption/ Decryption

- Uses the same cryptographic key for both encryption of plaintext and decryption of ciphertext

"Three may keep a secret, if two of them are dead."
— Benjamin Franklin, Poor Richard's Almanack



Symmetric Key Encryption/Decryption



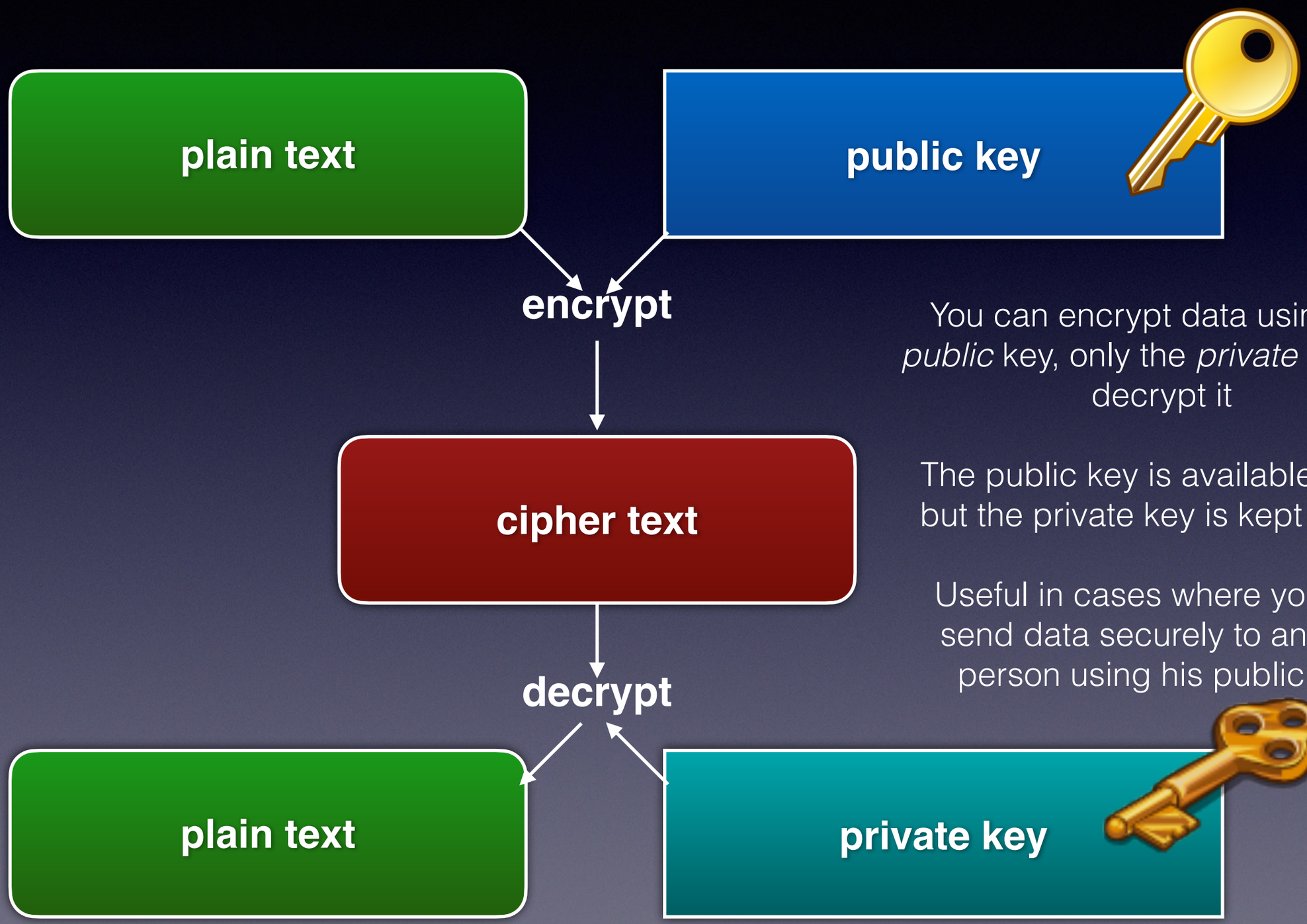
You can encrypt and decrypt data using the *shared* key.

Useful in cases where only one party knows the key and need not share the key with someone else

Asymmetric Encryption/ Decryption

- A public key algorithm generates two keys that are mathematically linked
  - one public and one private
    - You can encrypt data with a public key and decrypt with the private key
      - I encrypt my message with your public key and only you can decrypt with your private key
    - You can encrypt data with a private key and decrypt with the public key
      - I can verify that the message indeed comes from you
  - Examples of public key algorithms - RSA (Rivest–Shamir–Adleman)

Public-Key Cryptography

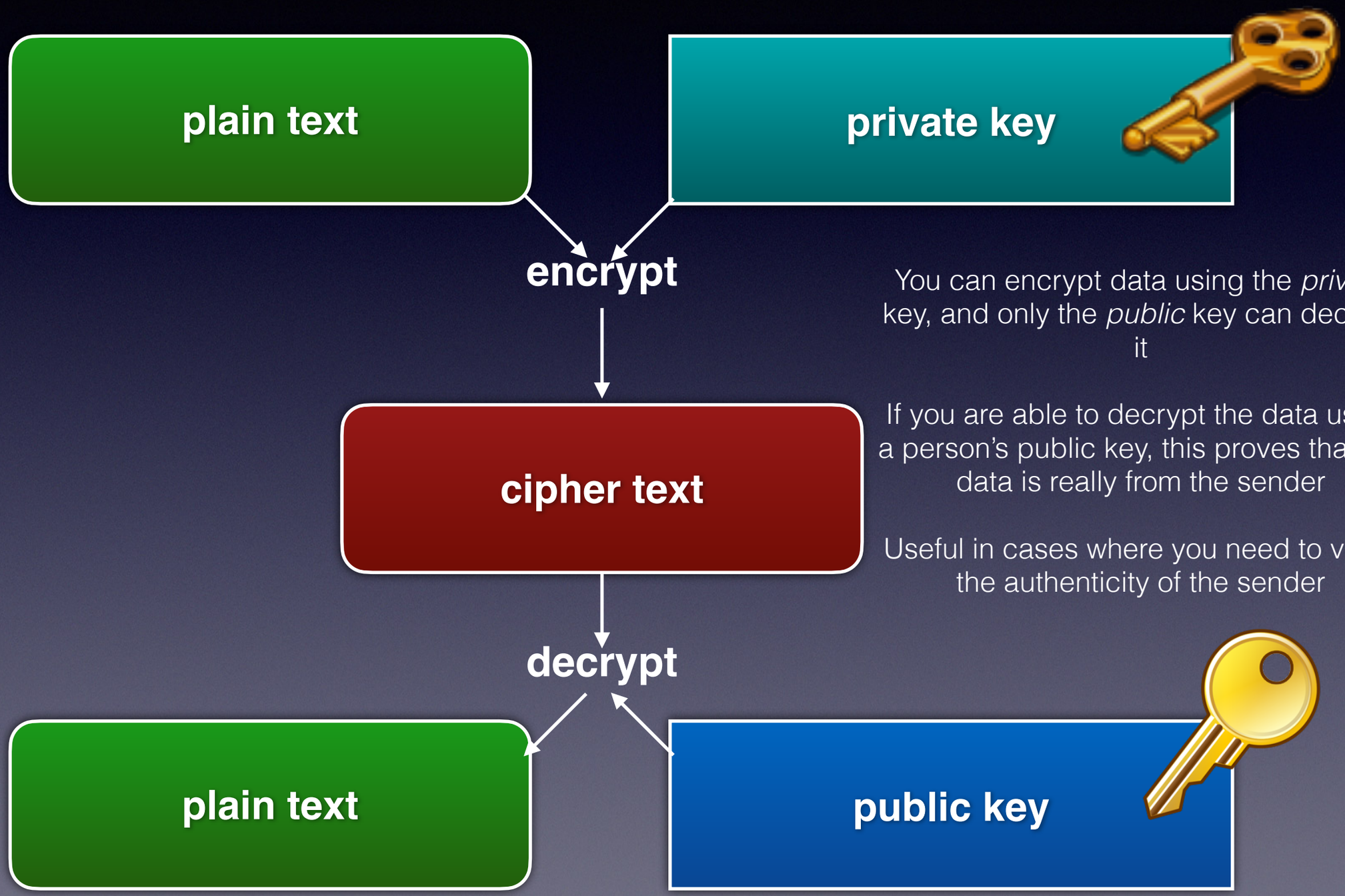


You can encrypt data using the *public* key, only the *private* key can decrypt it

The public key is available to all, but the private key is kept secret

Useful in cases where you can send data securely to another person using his public key

Public-Key Cryptography

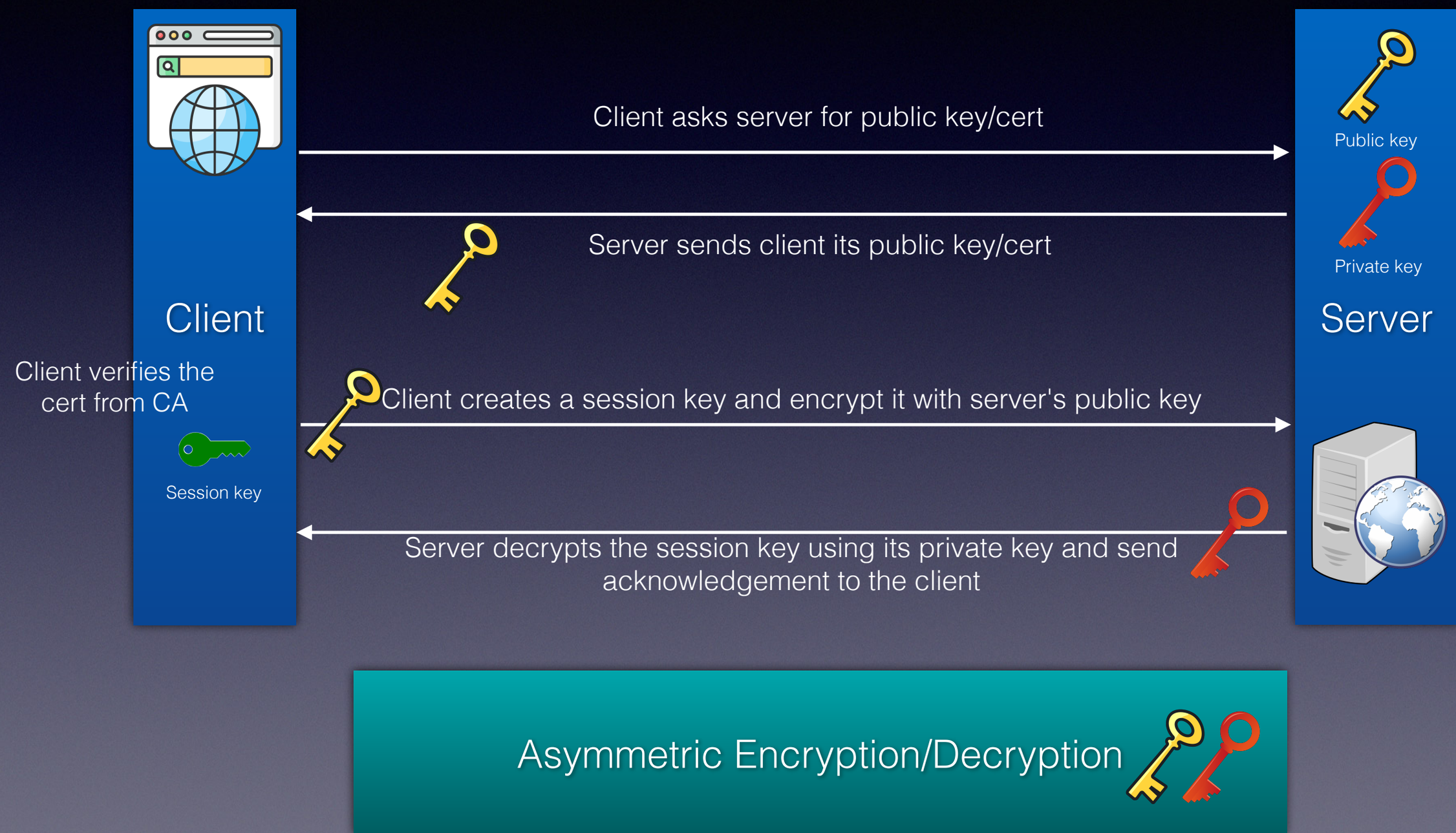


You can encrypt data using the *private* key, and only the *public* key can decrypt it

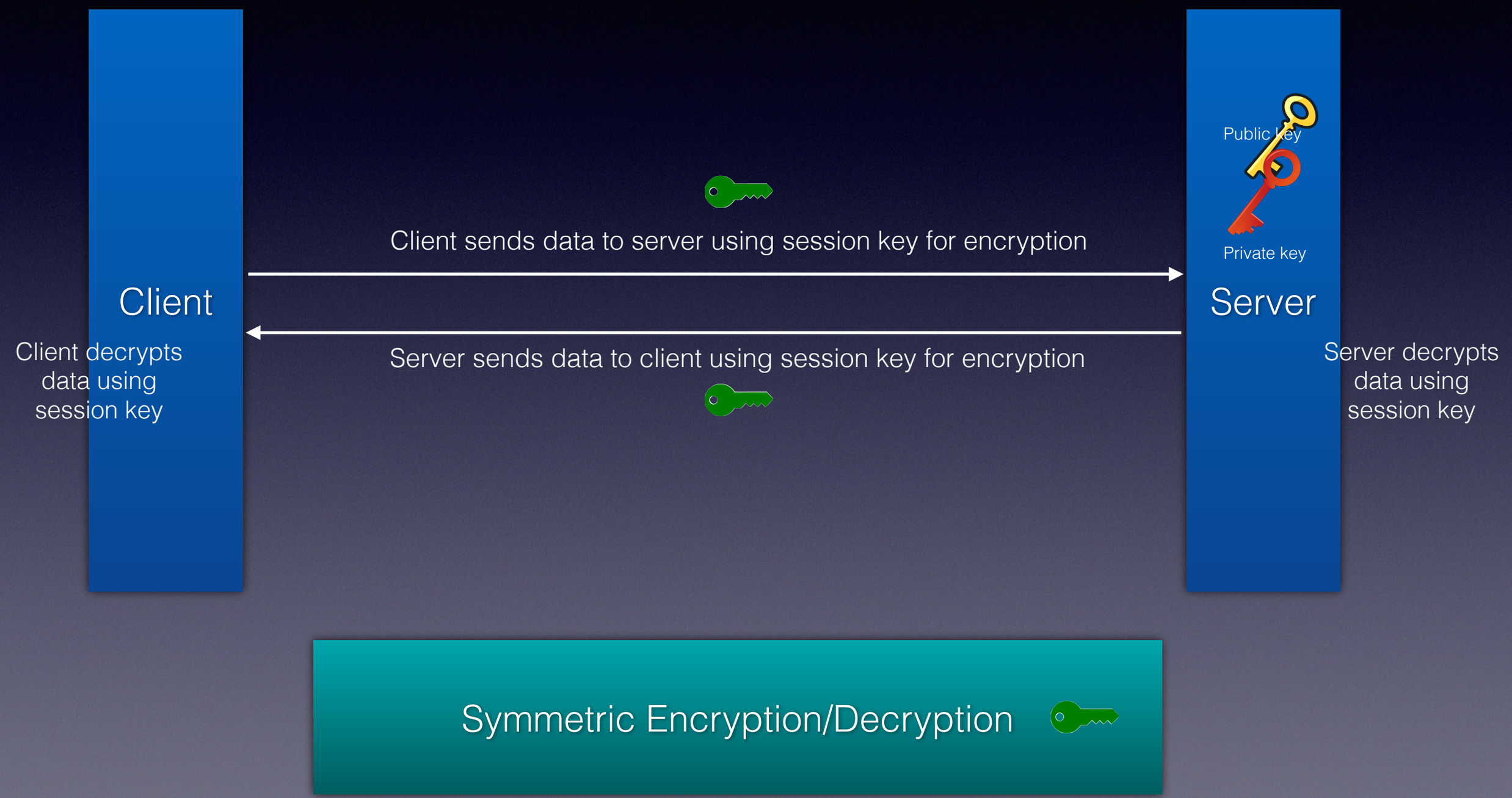
If you are able to decrypt the data using a person's public key, this proves that the data is really from the sender

Useful in cases where you need to verify the authenticity of the sender

SSL Handshake



Actual Data Transfer



(c) Wei-Meng Lee weimenglee@learn2develop.net

Exercises

Checking for Domain Availability

- The **python-whois** library allows you to retrieve WHOIS information for domain name
- Install the package:
  - `!pip install python-whois`

```
import whois
import sys
try:
    domain = whois.whois('Amazon.com')
    if domain.domain_name == None:
        sys.exit(1)
    print(domain)
except:
    print('Domain is available')
else:
    print('Domain already taken')
```

Getting your own IP address

- Using the socket module, you can get the current machine's hostname and IP address

```
import socket
host = socket.gethostname()
ip = socket.gethostbyname(host)
print(host, ip)
```

Getting the IP address of a domain

- Get the IP address of Google

```
try:
    print(socket.gethostbyname('www.google.com'))
except socket.error as err:
    print(err)
```

Getting all IP addresses associated with all network interfaces

- The **psutil** library provides more detailed network interface information

```
import psutil

addrs = psutil.net_if_addrs()
ip_addresses = []

for interface, addr_list in addrs.items():
    for addr in addr_list:
        if addr.family == socket.AF_INET:
            ip_addresses.append((interface, addr.address))

print("IP Addresses:", ip_addresses)
```

Finding the service at each port

- Output a list of TCP ports and their associated service names

```
import socket

services = {}

def find_service_name():
    protocolname = 'tcp'
    for port in range(1024):
        try:
            service_name = socket.getservbyport(port, protocolname)
            services[port] = service_name
        except OSError:
            # Skipping ports that do not have an associated service name
            continue

    # Print the results after collecting all service names
    for port, service_name in services.items():
        print(f"Port: {port} => {service_name}")

find_service_name()
```

Find port by service name

```
{service:port for port,service in
    services.items()}['smtp']
```

```
{service:port for port,service in
    services.items()}['https']
```

Find a free unused port

- You can find a random port that is unused

```
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    print(s.bind(('localhost', 0))) # use 0 to generate a random port
    print(s.getsockname())
```

Socket Programming

- Using Python, create a:
  - Socket Server
    - allows multiple clients to connect to it
    - broadcast received message to all connected clients except the client who sent the message
  - Socket Client
  - Allow your server to be accessible from outside the VM
  - Use port 8080

Download Source

- https://bit.ly/3YYOdHz

Server

```
import socket                          # import the socket module
import _thread

client_list = []                       # stores all the clients connected to the server

def on_new_client(clientsocket,addr):  # addr is the address of the client
    while True:                        # always listen for incoming messages from the client
        msg = clientsocket.recv(1024)
        if msg:
            print("Sending data back to the client...")
            print("Received from client: " + msg.decode())
            for client in client_list:
                if client[0] != clientsocket:     # don't send back to the client who sent the message
                    # broadcast the message to other client
                    print('Sending message to ',client[1])
                    client[0].send(str.encode(f'{addr} > {msg.decode()}'))
        else:
            print ("Client has disconnected.")
            break
    clientsocket.close()

s = socket.socket()                    # Create a socket object
host = socket.gethostbyname("")        # Get local machine name
port = 8080                            # port to connect

print("Server started!")
print("Waiting for clients...")

s.bind((host, port))                   # Bind to the port
s.listen(5)                            # wait for client connection

while True:
    c, addr = s.accept()               # Establish connection with client
    client_info = [c, addr]            # save the client details into the list
    client_list.append(client_info)
    print("Connected from: ", addr)

    # spin off a new thread to handle the client
    _thread.start_new_thread(on_new_client,(c,addr))

s.close()
```

Client

```
import socket
import _thread

def on_receive_msg(s):
    while True:
        print(s.recv(1024).decode())

s = socket.socket()
host = socket.gethostbyname("x.x.x.x")                # ip address of server
port = 8080

s.connect((host, port))                          # connect to the server
_thread.start_new_thread(on_receive_msg,(s,))  # listen from server

while True:
    toSend = input()
    s.send(toSend.encode())

s.close()
```

Note

- If server is on a VM, you need to connect the client to the server using its public IP address.
  - whatismyip.com
- You also need to configure Windows Defender Firewall to let the port 8080 for inbound and outbound traffic

Requests Library

- The **requests** library in Python is a simple and elegant HTTP library for making HTTP requests.

```
import requests
import json

def get_request():
    response = requests.get('https://jsonplaceholder.typicode.com/todos')
    if response.status_code == 200:
        print("Response from server:")
        print(response.json())

        users = response.json()
        for user in users:
            print(user['id'])
            print('====')
            print(user['title'])
    else:
        print(f"Failed to retrieve data. Status code: {response.status_code}")

get_request()
```

Web Scraping using BeautifulSoup

- BeautifulSoup is a popular Python library used for web scraping and parsing HTML and XML documents.

```
import requests
from bs4 import BeautifulSoup

def web_scraping():
    url = 'https://www.facebook.com'
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        title = soup.find('title').text
        print(f"Title of the page: {title}")
    else:
        print(f"Failed to retrieve the page. Status code: {response.status_code}")

if __name__ == "__main__":
    web_scraping()
```

```python
from bs4 import BeautifulSoup

html_doc = """
<html>
    <head><title>The Dormouse's story</title></head>
    <body>
        <p class="title"><b>The Dormouse's story</b></p>
        <p class="story">Once upon a time there were three little sisters; and their names were
            <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
            <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
            <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
            and they lived at the bottom of a well.</p>
        <p class="story">...</p>
    </body>
</html>
"""

soup = BeautifulSoup(html_doc, 'html.parser')

print(soup.prettify())
print("Title of the page:", soup.title.string)
print("First paragraph:", soup.p)
print("All paragraphs:", soup.find_all('p'))
print("All links:", soup.find_all('a'))
print("Link 1:", soup.find(id='link1'))
```