Network Protocols and Trust

Logan Prough | 08 December 2022 | OISF

Disclaimer

- Any views expressed in or during this presentation are solely those of the presenter, and do not represent the views of any other person or organization
- This presentation is provided for educational purposes only
- You are ultimately responsible for your own actions; don’t do anything you are not authorized to do

Introduction

- These materials were adapted from 8 hours worth of lecture material and 4 hours of hands-on lab exercises
- The original materials targeted an audience of college undergraduates
  - The content may be review for many audience members
- Please feel free to ask questions at any time
- Deeper discussion on any covered topic is highly encouraged

Overview

- Networking and OSI model review
- Network protocols, trust models, and attacks

- ARP	- DNS	- TLS
- DHCP	- NTP	- HTTP(S)

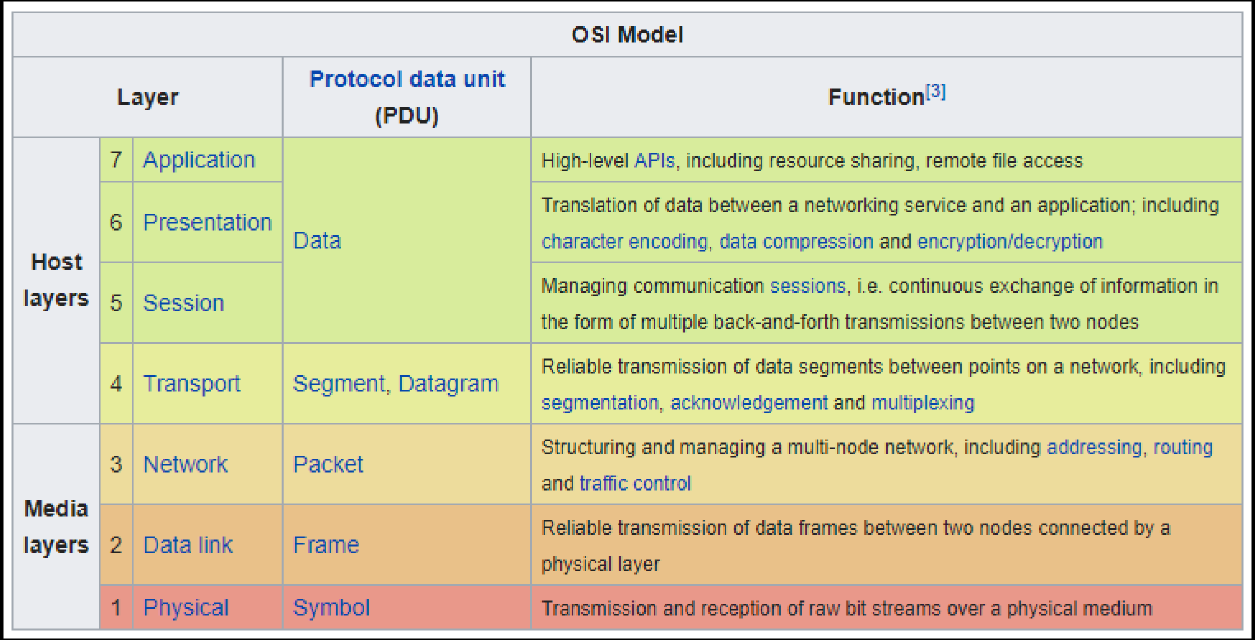
- Discussion

“The world is a jungle in general, and the networking game contributes many animals.”

- David C. Plummer, RFC 826

Open Systems Interconnection (OSI) Model

- Logically separates network functions into seven layers
- This presentation mostly focuses on layers 1-4
- Frame: Layer 2
- Packet: Layer 3
- Which layer handles authentication?



OSI Model

Layer			Protocol data unit (PDU)	Function[3]
Host layers	7	Application	Data	High-level APIs, including resource sharing, remote file access
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1	Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

Image source: https://en.wikipedia.org/wiki/OSI_model

Network Interface Controller (NIC)

- Contains hardware to connect a computer to a network
- Examples: wired (Ethernet) or wireless (WiFi)
- An operating system sends data to a NIC, which transmits it over some physical medium
- The NIC receives data on the physical medium and passes it to the operating system

Media Access Control (MAC) Address

- Layer 2 in the OSI model
- 48-bit address which “uniquely” identifies a NIC
- Usually represented as six colon-separated hexadecimal octets
- Used to address frames sent to a destination within the same broadcast domain
- Example MAC address: **34:e6:d7:21:bc:97**

Internet Protocol (IP) Address

- Layer 3 in the OSI model
- Address assigned to a network interface on an IP network
- IPv4 uses 32-bit addresses
- IPv6 uses 128-bit addresses
- Divided into network bits and host bits
  - Network bits identify the subnet an address belongs to
  - Host bits identify an individual host on the subnet
- The address with all host bits set to 1 represents the broadcast address for the given subnet
- The address with all host bits set to 0 is generally considered invalid

Example IP Addresses

IPv4 address: **10.50.32.200**

IPv6 address:
2001:0db8:0a0b:12f0:0000:0000:0000:0001

The same IPv6 address, short notation:
2001:db8:a0b:12f0::1

IPv4 Address Breakdown

- Consider the IP address **10.50.32.200/24**
- **/24** is Classless Inter-Domain Routing (CIDR) notation for the subnet mask 255.255.255.0
- The first 24 bits of the address (10.50.32) represent the subnet
- The remaining 8 bits (.200) identify the host on the subnet

	Network bits						Host bits
Decimal representation:	**10**	**.**	**50**	**.**	**32**	**.**	**200**
Binary representation:	00001010		00110010		00100000		11001000

OSI Model Example

- Layer 1: Copper cable (e.g. cat 6a)
- Layer 2: IEEE 802.3 Ethernet
  - Frame header contains*
    - Destination MAC address
    - Source MAC address
- Layer 3: IP
  - Packet header contains*:
    - Source IP address
    - Destination IP address

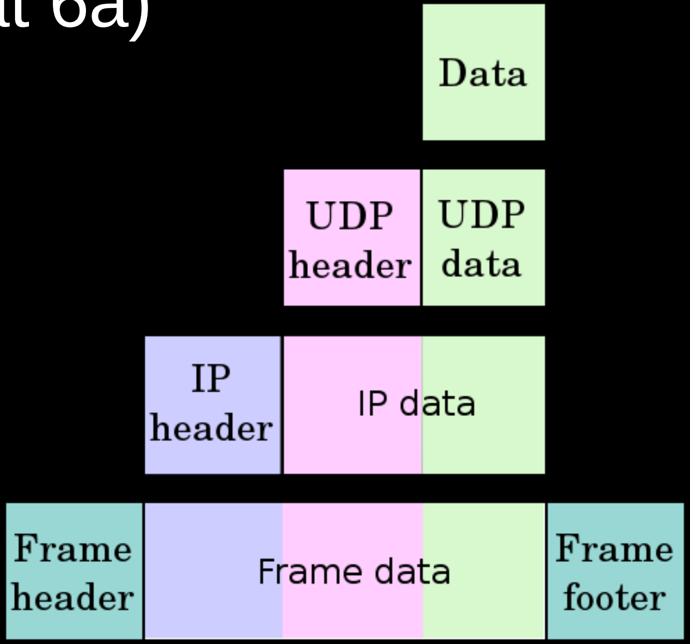


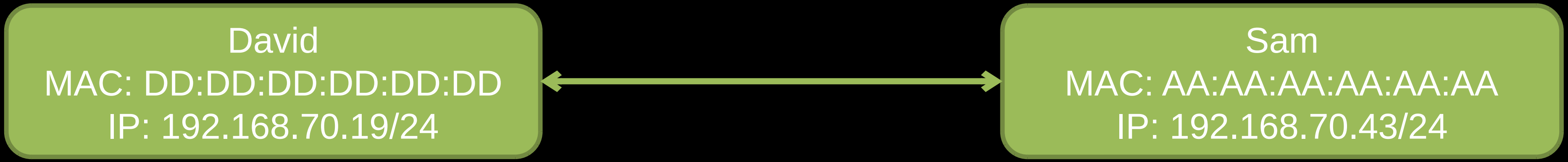
Image source: https://en.wikipedia.org/wiki/File:UDP_encapsulation.svg

*among other things

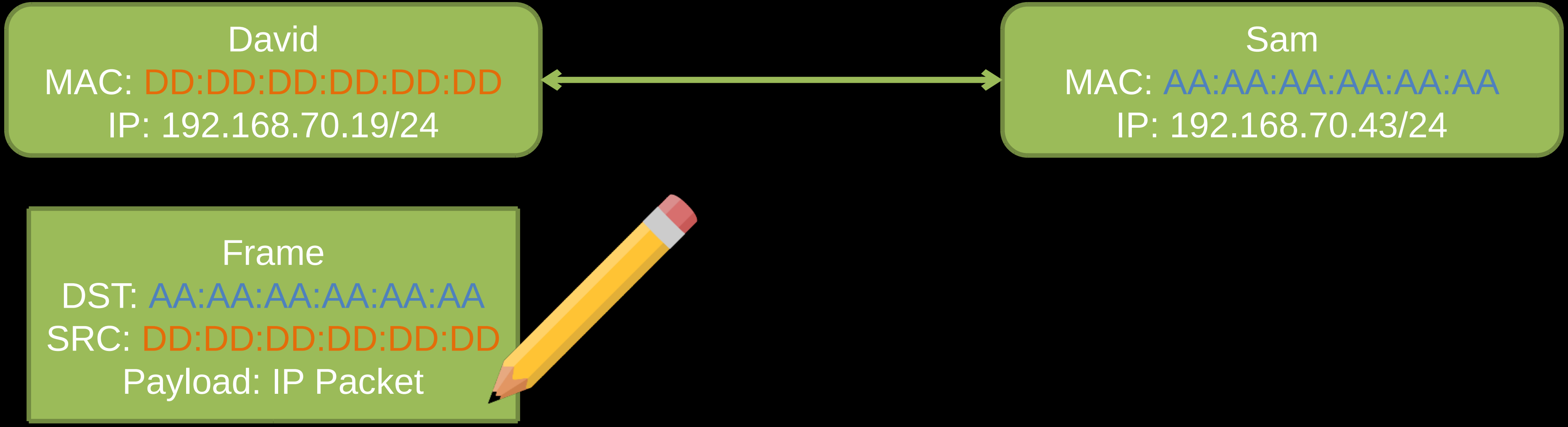
OSI Model Example

- Layer 4: User Datagram Protocol (UDP)
  - Designed as a stateless, lightweight protocol
  - Does not guarantee reliability or ordering
  - Datagrams include a checksum for integrity
- Layer 4: Transmission Control Protocol (TCP)
  - Provides a stream of data, often bidirectional
  - Aims to deliver data reliably, in order, with checksum integrity
  - Requires 3-way handshake to establish connection
- Both TCP and UDP introduce port numbers

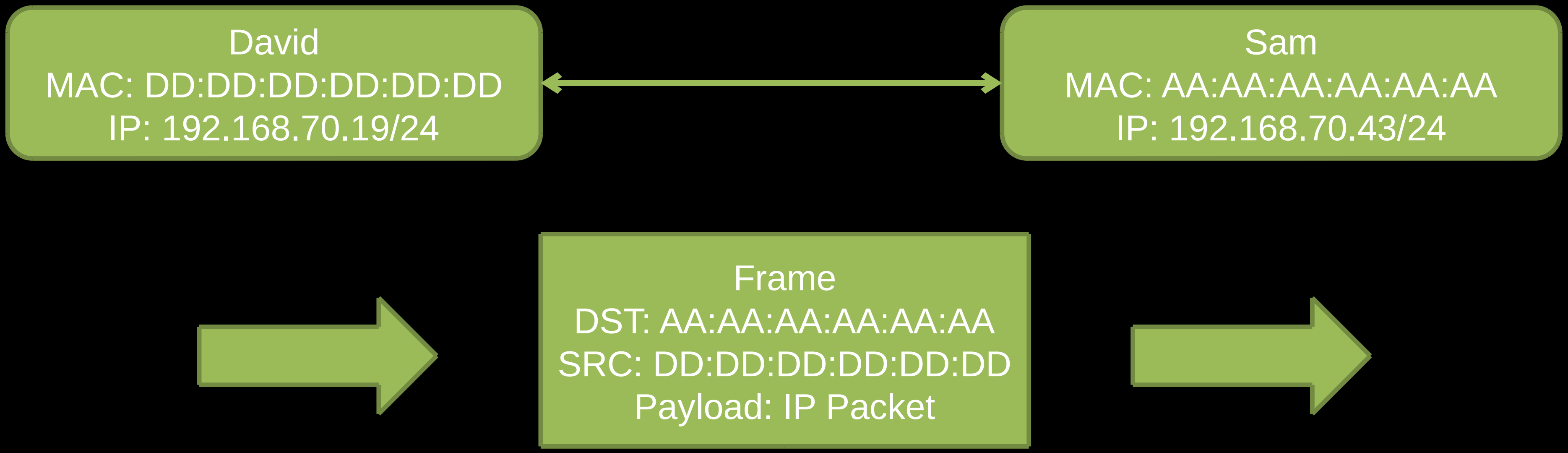
Layer 2 Example



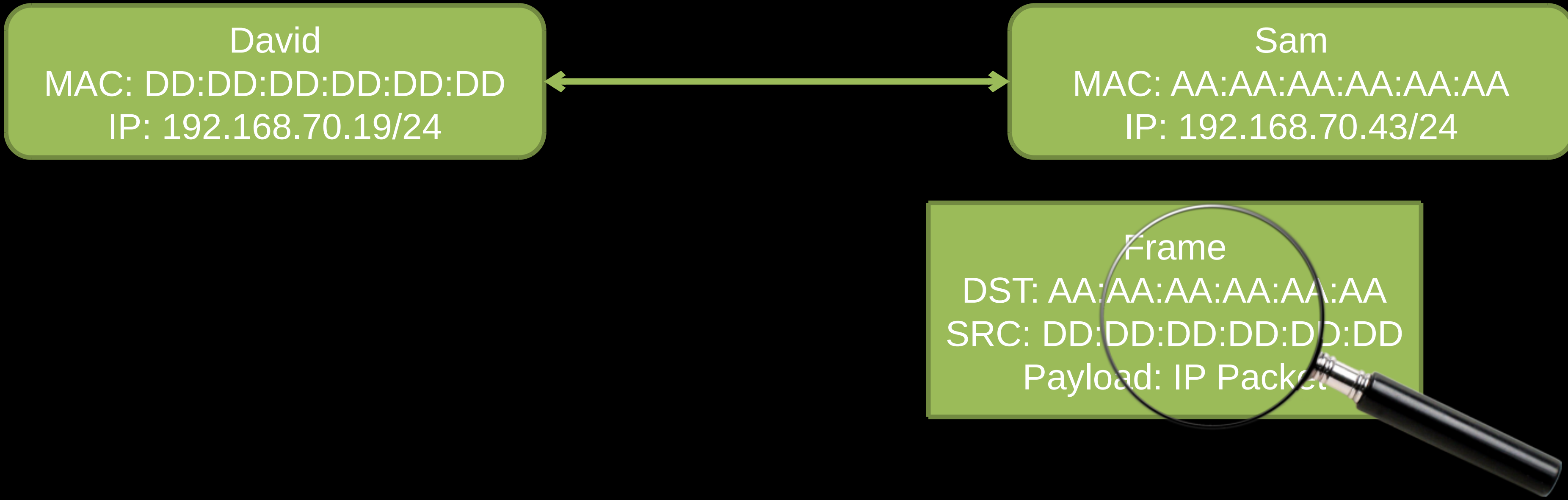
Layer 2 Example



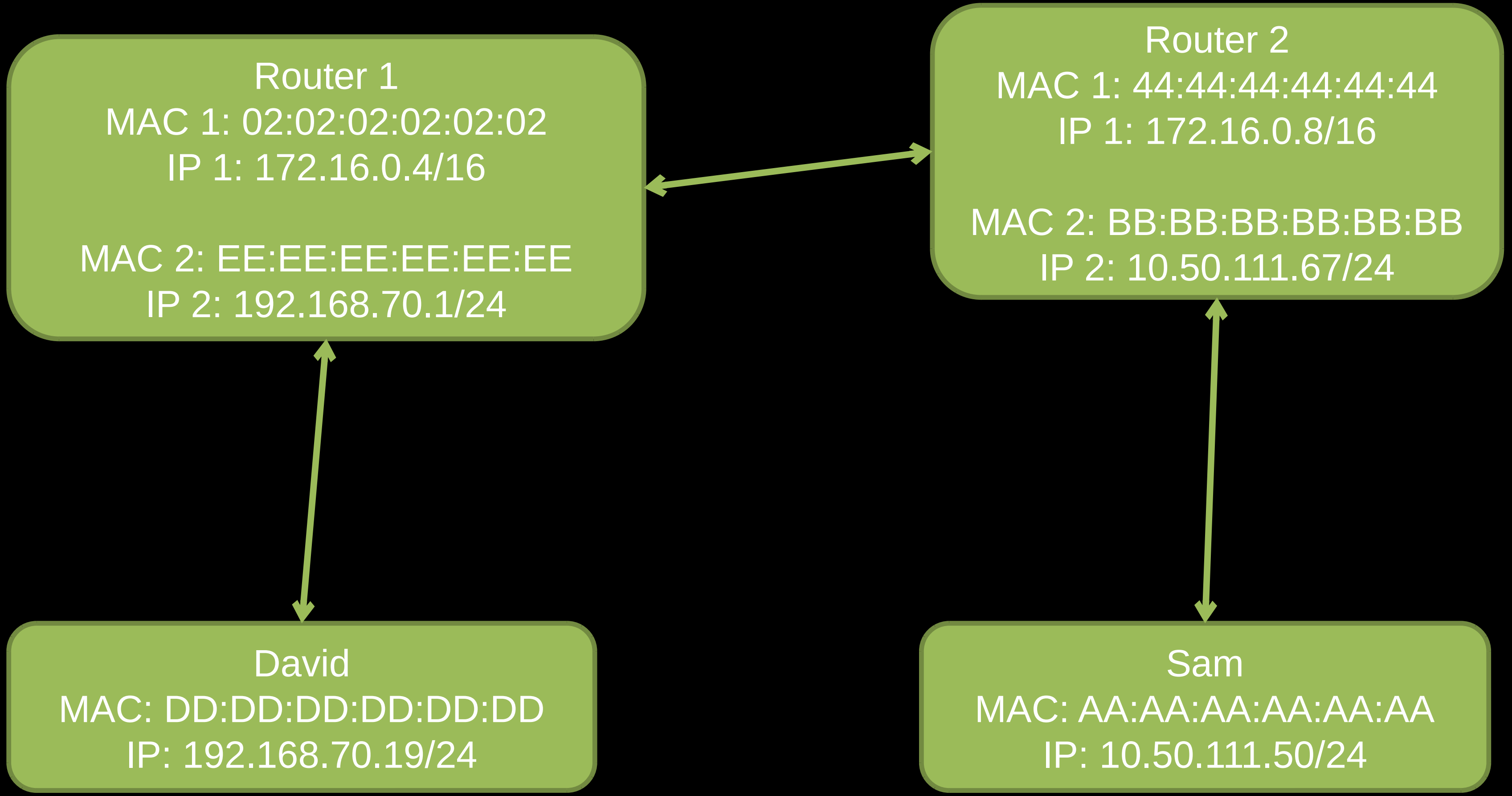
Layer 2 Example



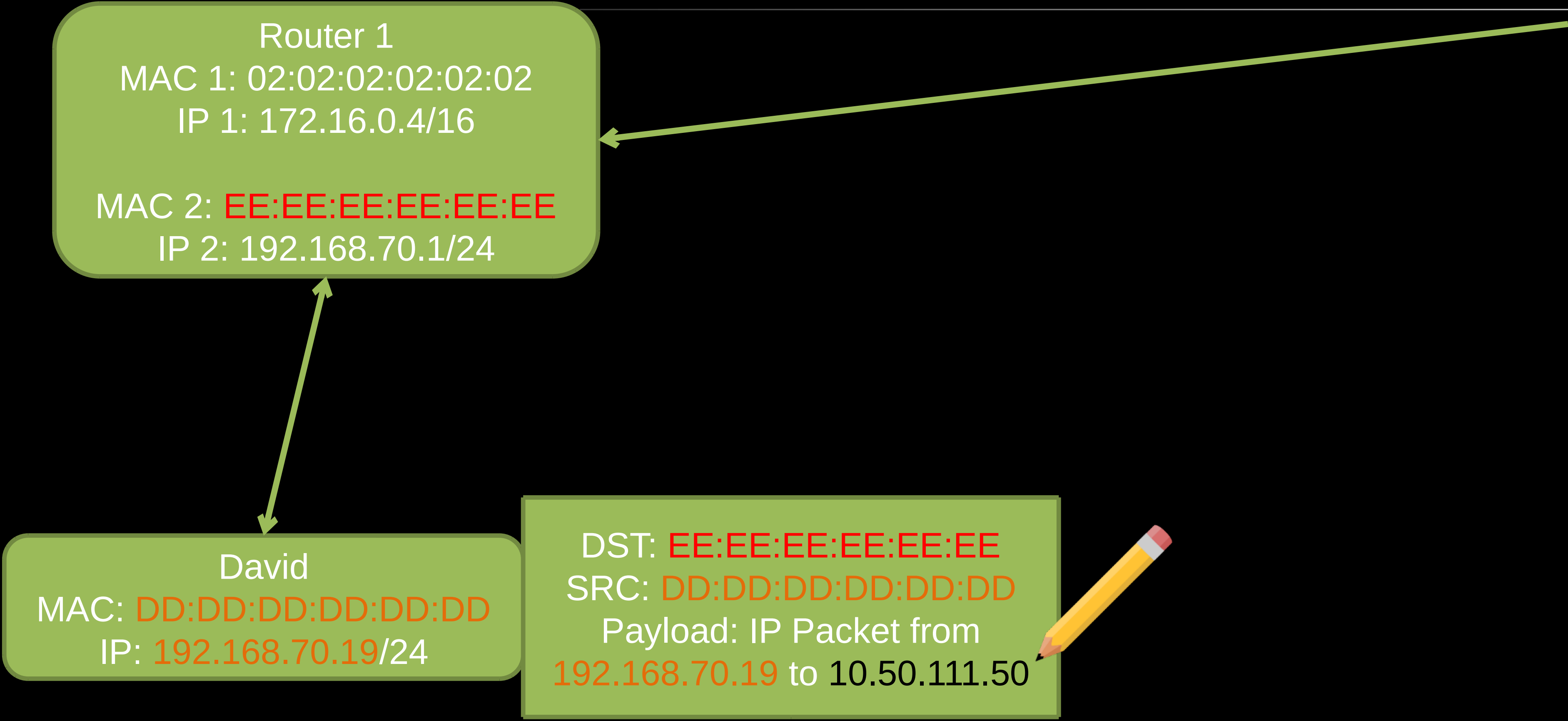
Layer 2 Example



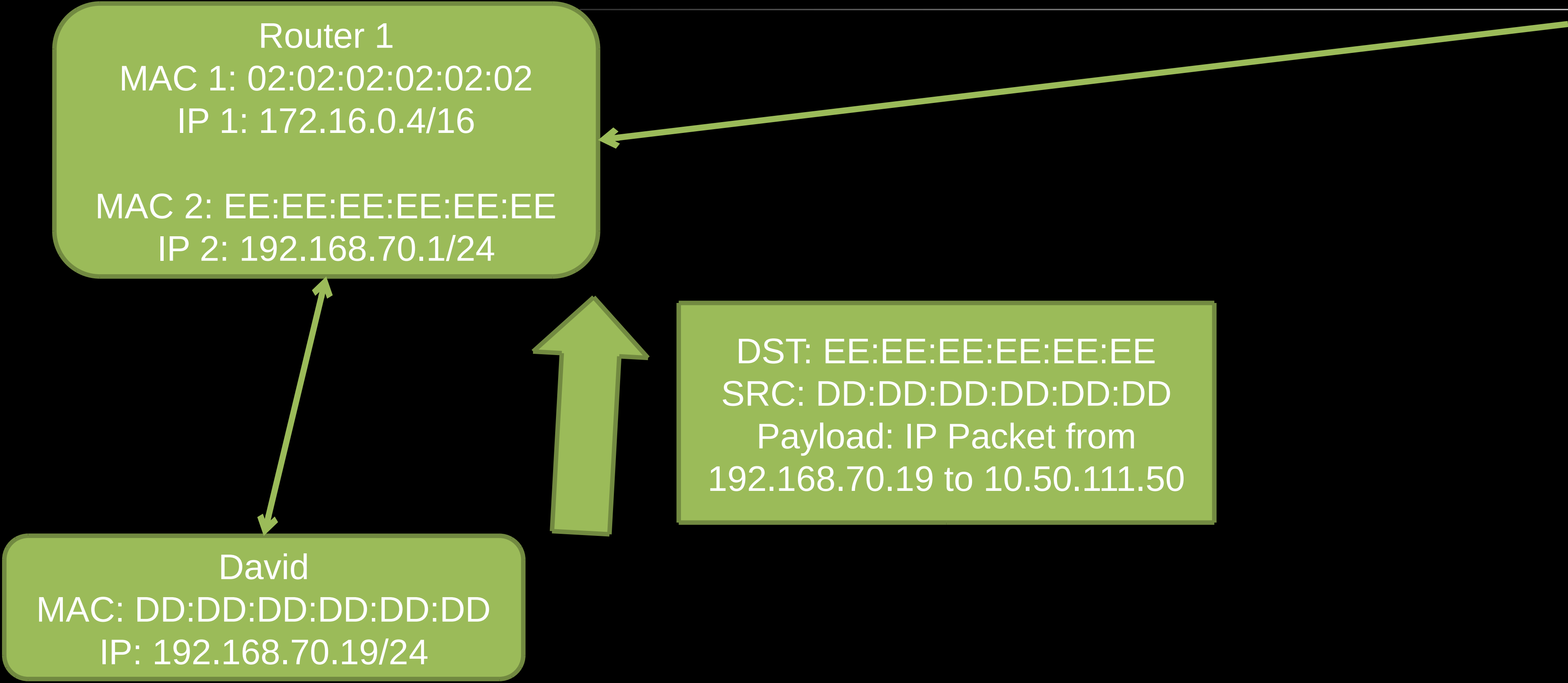
Layer 3 Example



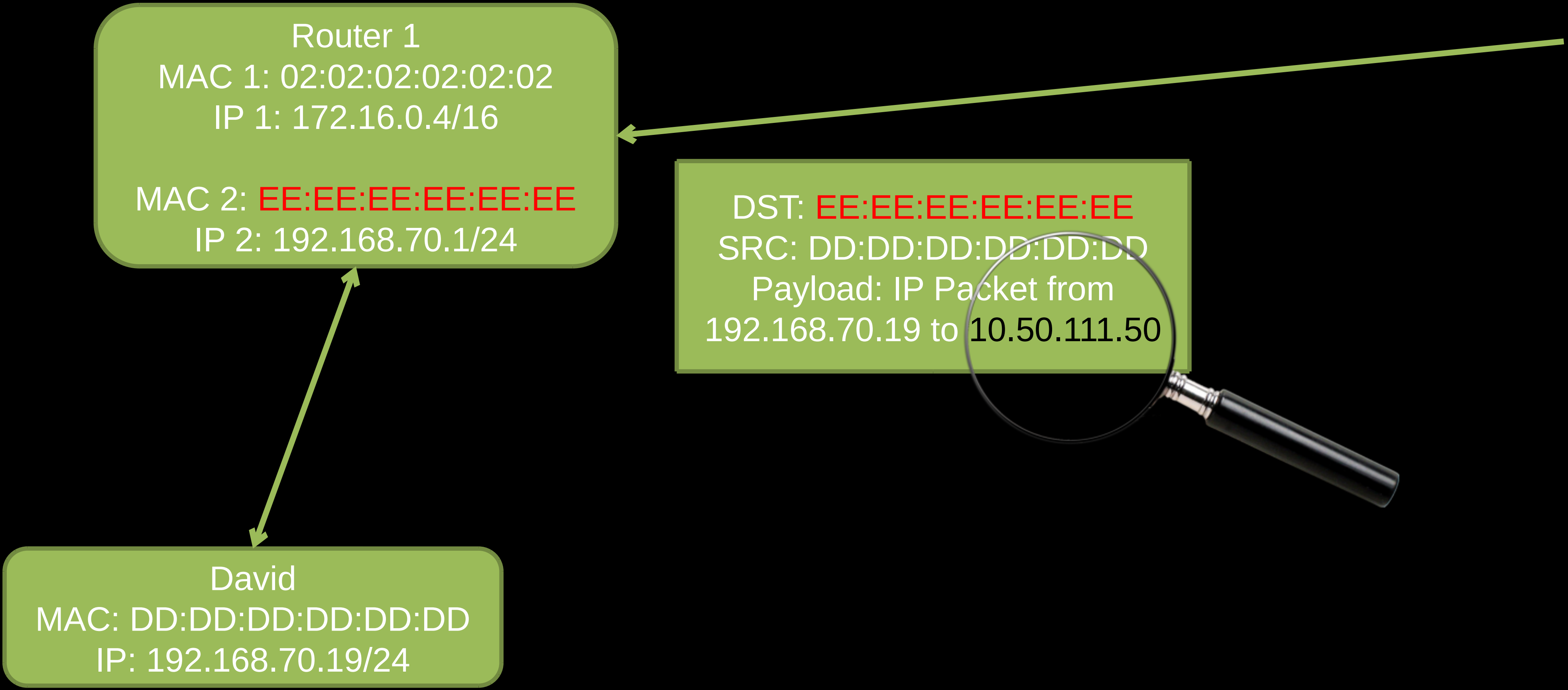
Layer 3 Example



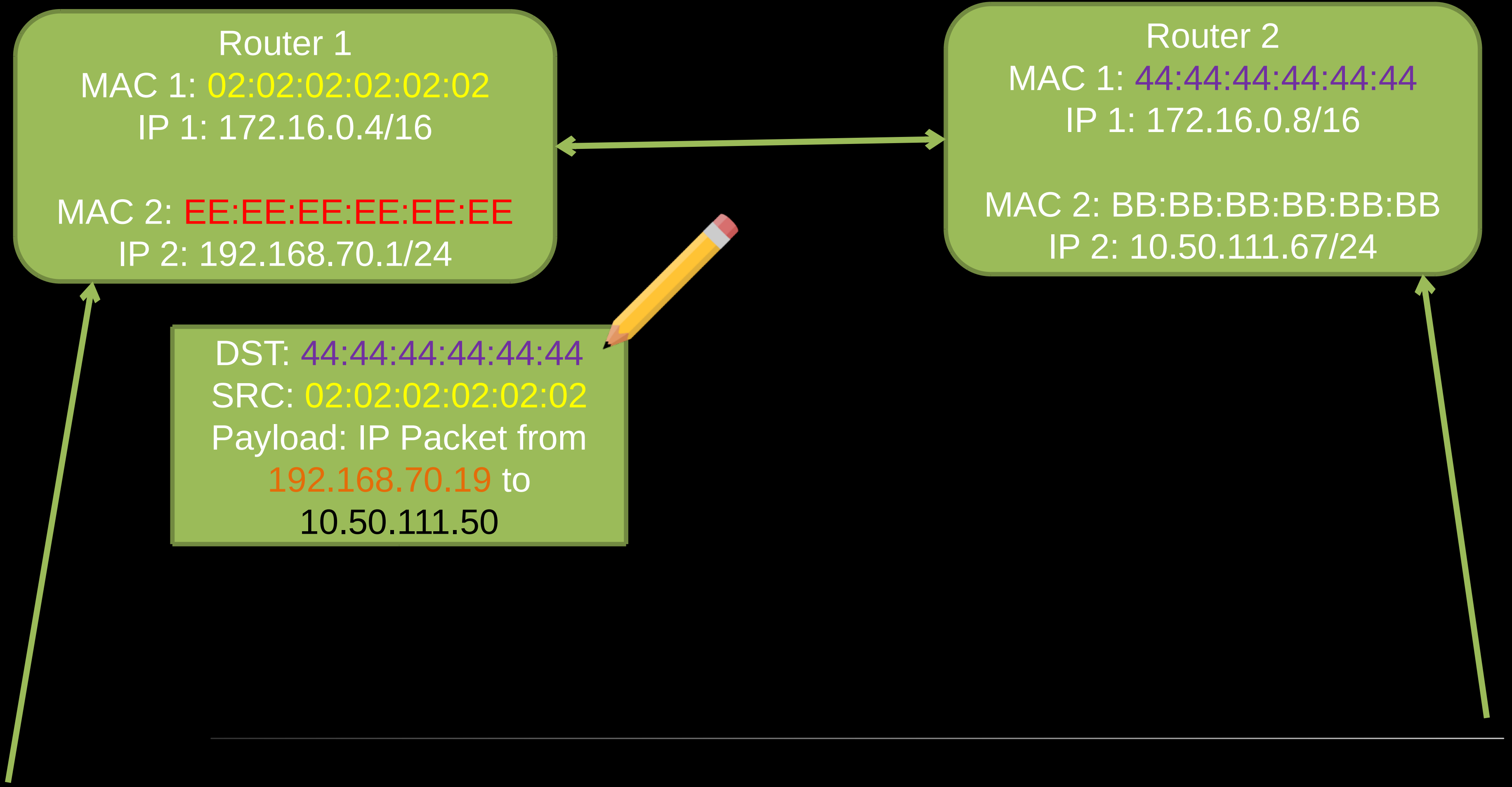
Layer 3 Example



Layer 3 Example



Layer 3 Example



Router 1
MAC 1: 02:02:02:02:02:02
IP 1: 172.16.0.4/16

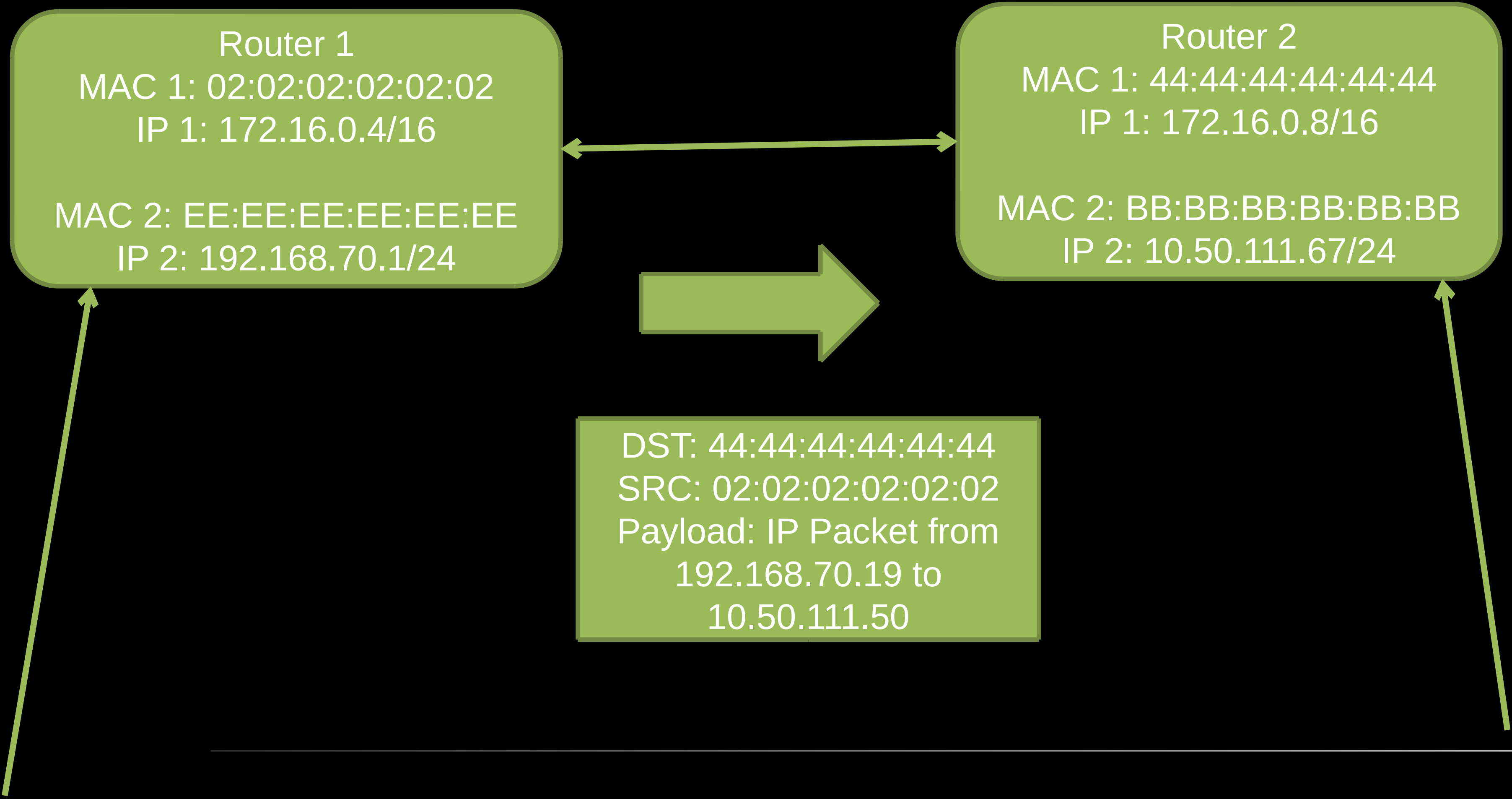
MAC 2: EE:EE:EE:EE:EE:EE
IP 2: 192.168.70.1/24

Router 2
MAC 1: 44:44:44:44:44:44
IP 1: 172.16.0.8/16

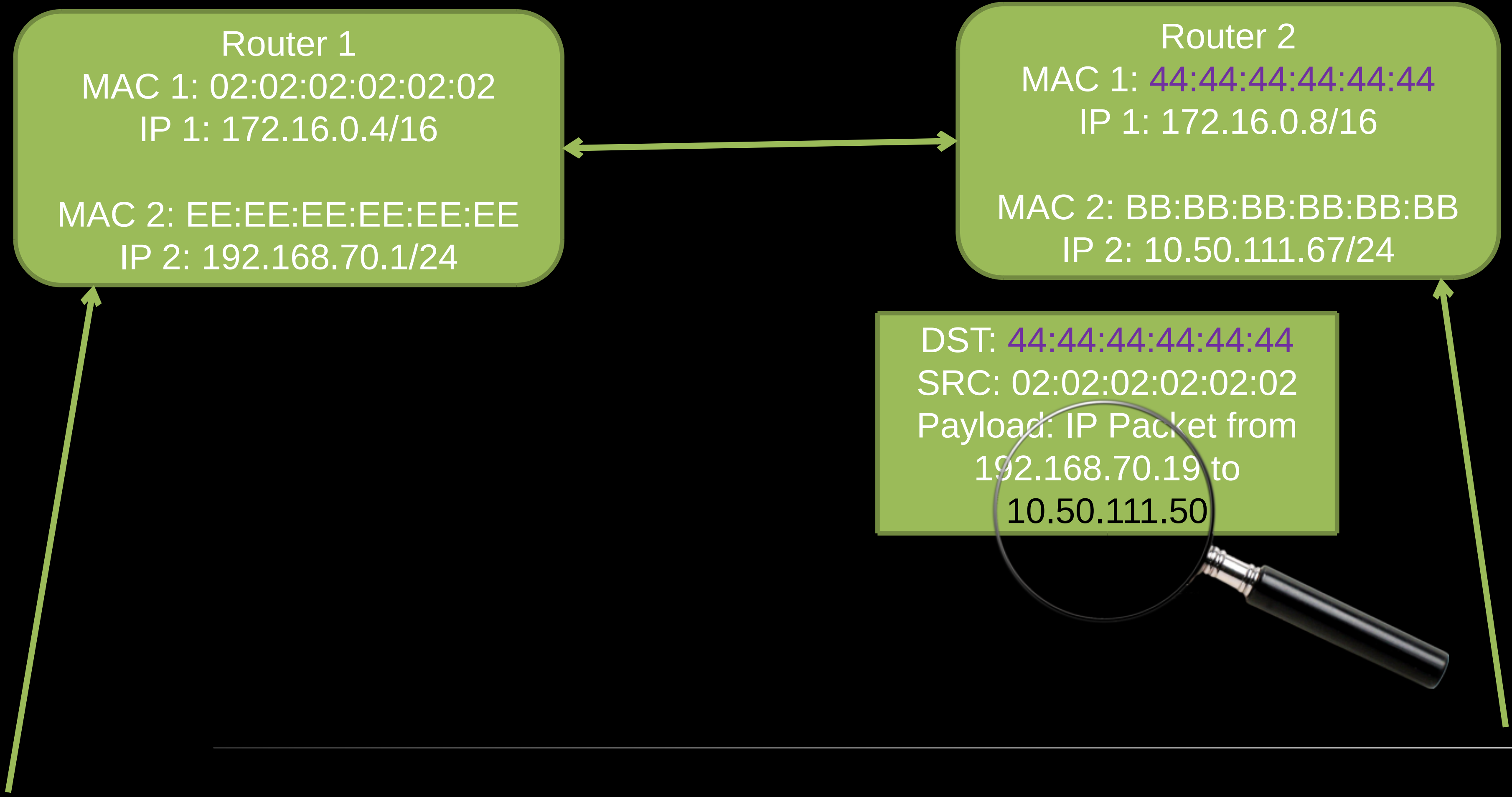
MAC 2: BB:BB:BB:BB:BB:BB
IP 2: 10.50.111.67/24

DST: 44:44:44:44:44:44
SRC: 02:02:02:02:02:02
Payload: IP Packet from 192.168.70.19 to 10.50.111.50

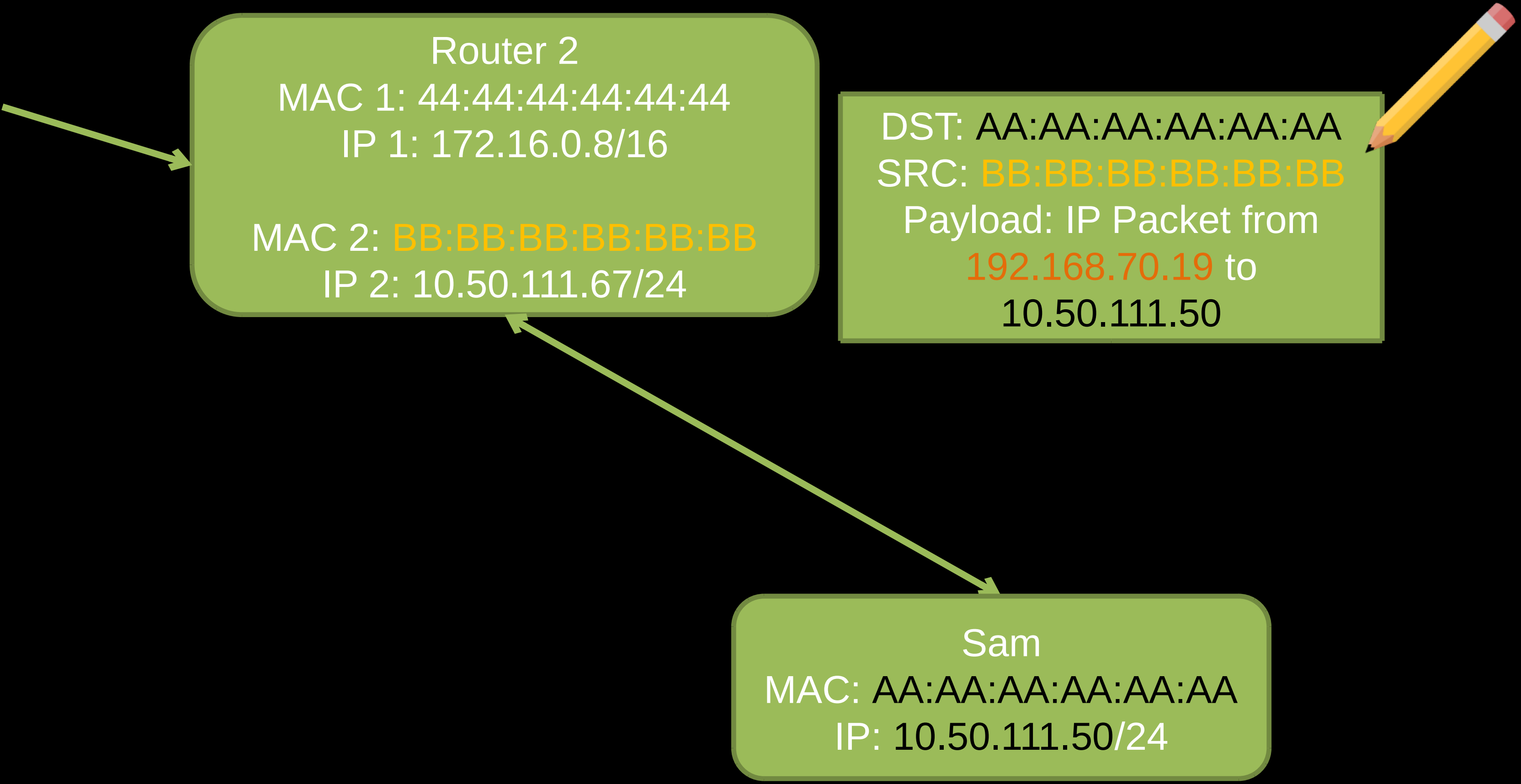
Layer 3 Example



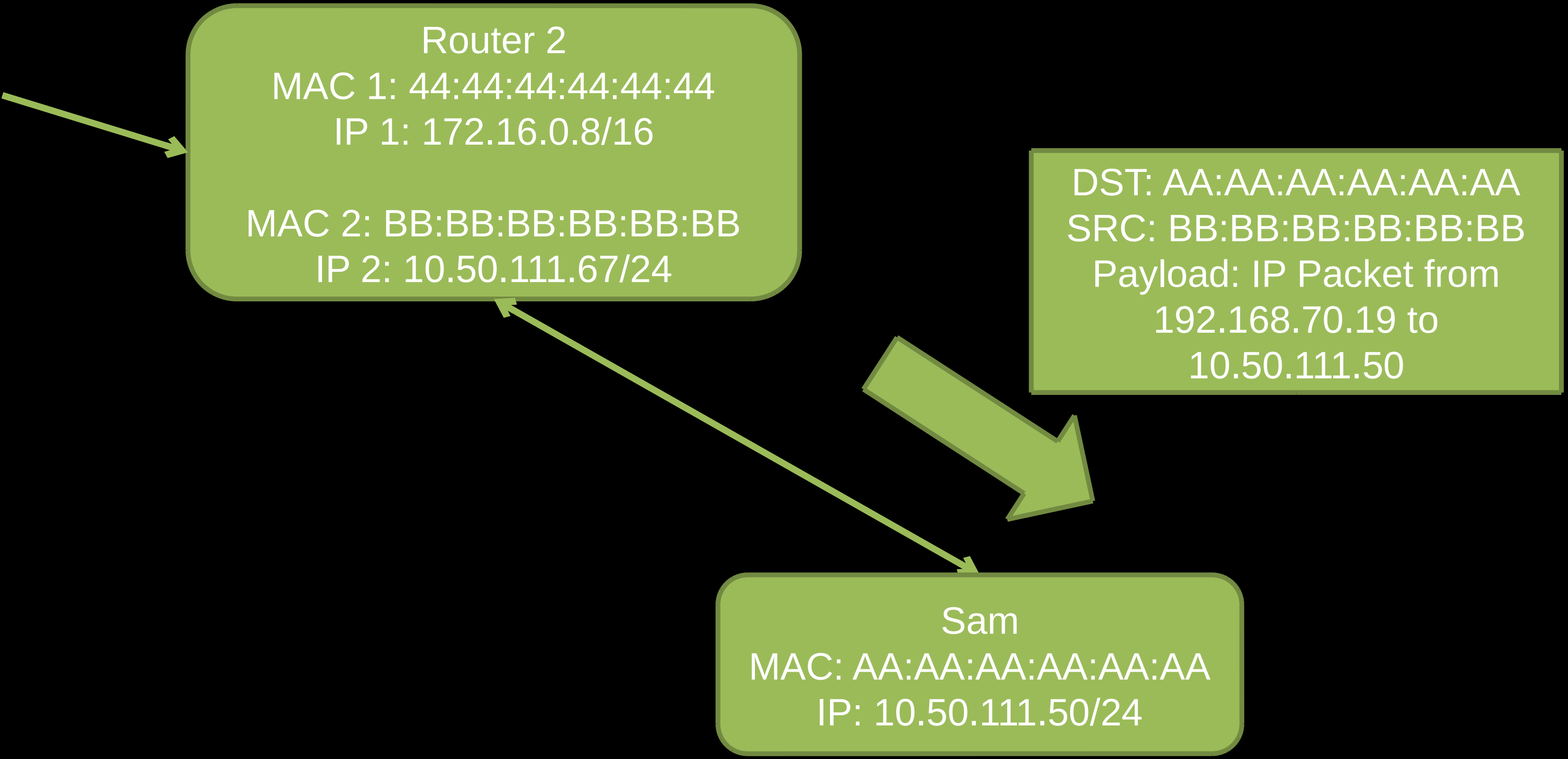
Layer 3 Example



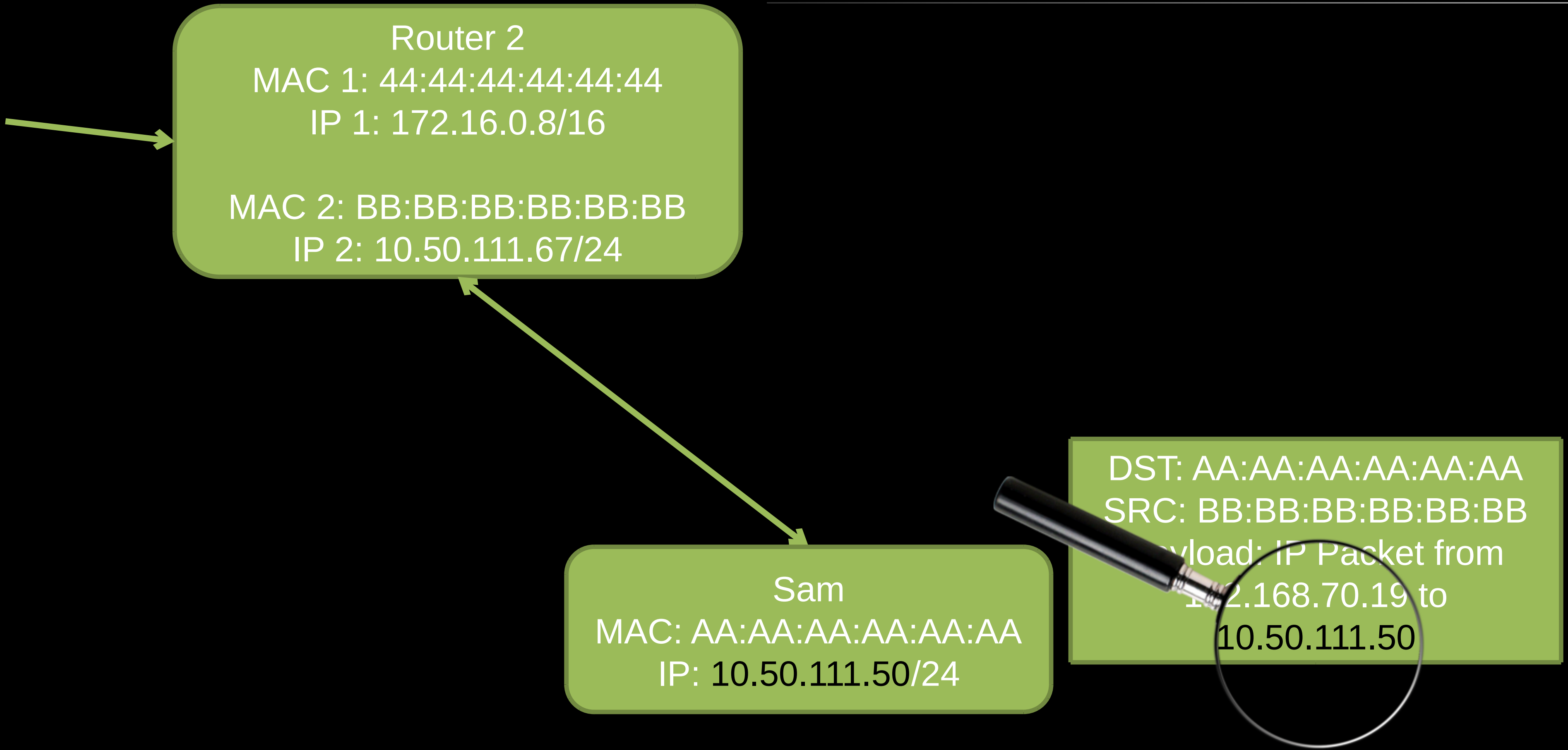
Layer 3 Example



Layer 3 Example



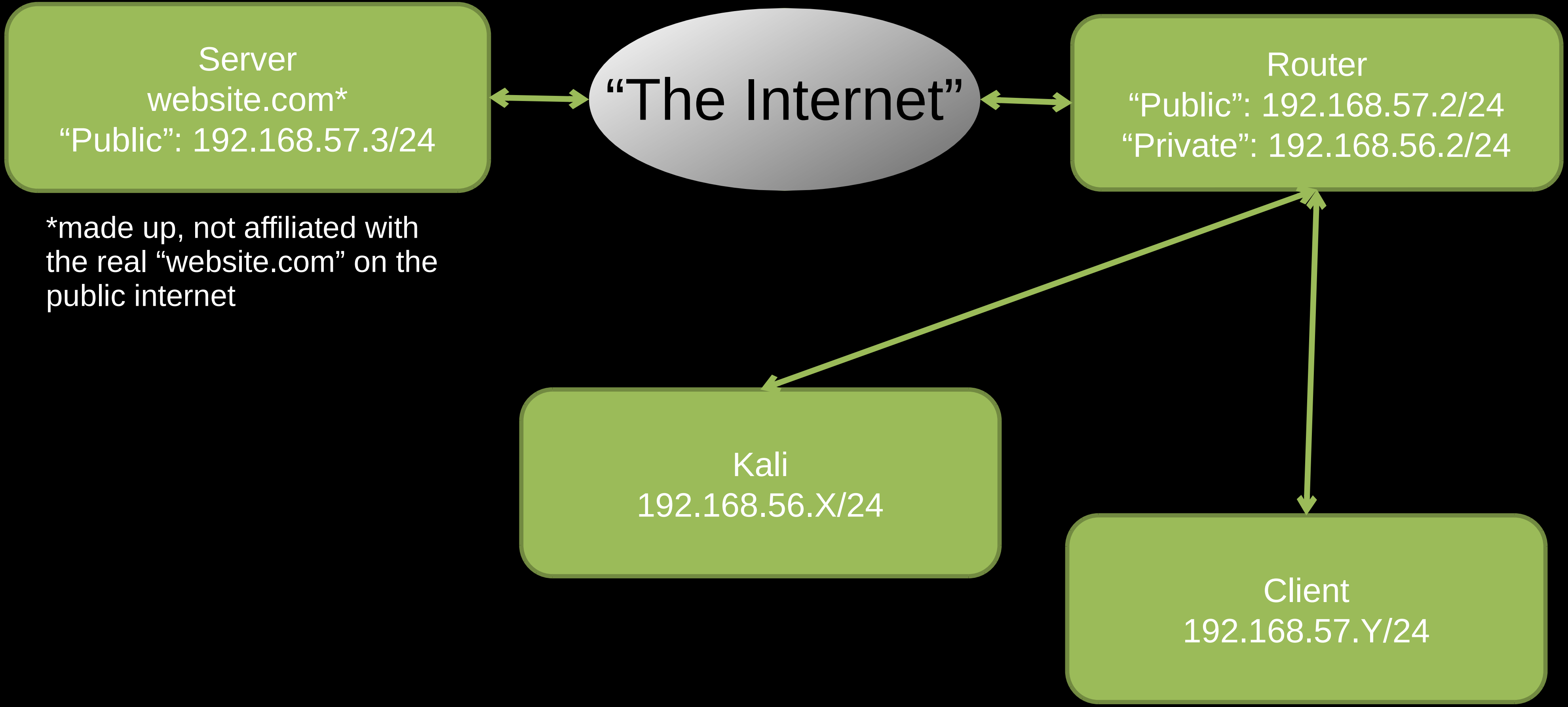
Layer 3 Example



This was an oversimplification

- How did David know he could not reach Sam within his broadcast domain (i.e. directly connected)?
  - David used his subnet mask to derive the network bits of his IP address, then compared the network bits of his IP address with Sam’s IP address and determined the network bits differed
- How did David know Sam’s IP address?
- How did David know Router 1’s MAC address?
- How did David know to send his packet to Router 1 in order for it to get to Sam?

Demo Setup



*made up, not affiliated with the real “website.com” on the public internet

Overview

- Networking and OSI model review
- **Network protocols, trust models, and attacks**

  - ARP **- DNS** **- TLS**

  - DHCP **- NTP** **- HTTP(S)**

- Discussion

Address Resolution Protocol (ARP)

- ARP connects layers 2 and 3 of the OSI model by allowing discovery of the link-layer address (MAC address) associated with a network layer address (IPv4 address)
- Example: imagine 192.168.70.4/24 wishes to send a packet to 192.168.70.7/24
  - 192.168.70.4 broadcasts the ARP request:
    - “Who has **192.168.70.7**? Tell **192.168.70.4**”
  - 192.168.70.7 sends the unicast reply:
    - “**192.168.70.7** is at **43:f8:c8:20:9a:ca**”
    - 192.168.70.7 already knows the MAC address of 192.168.70.4 because the frame header of the ARP request contains its source address

ARP Example

David
MAC: DD:DD:DD:DD:DD:DD
IP: 192.168.70.19/24

Sam
MAC: AA:AA:AA:AA:AA:AA
IP: 192.168.70.43/24

David's ARP Table

IP Address	MAC Address

Sam's ARP Table

IP Address	MAC Address

ARP Example

David
MAC: DD:DD:DD:DD:DD:DD
IP: 192.168.70.19/24

Sam
MAC: AA:AA:AA:AA:AA:AA
IP: 192.168.70.43/24

ARP Request: Who has 192.168.70.43? Tell 192.168.70.19

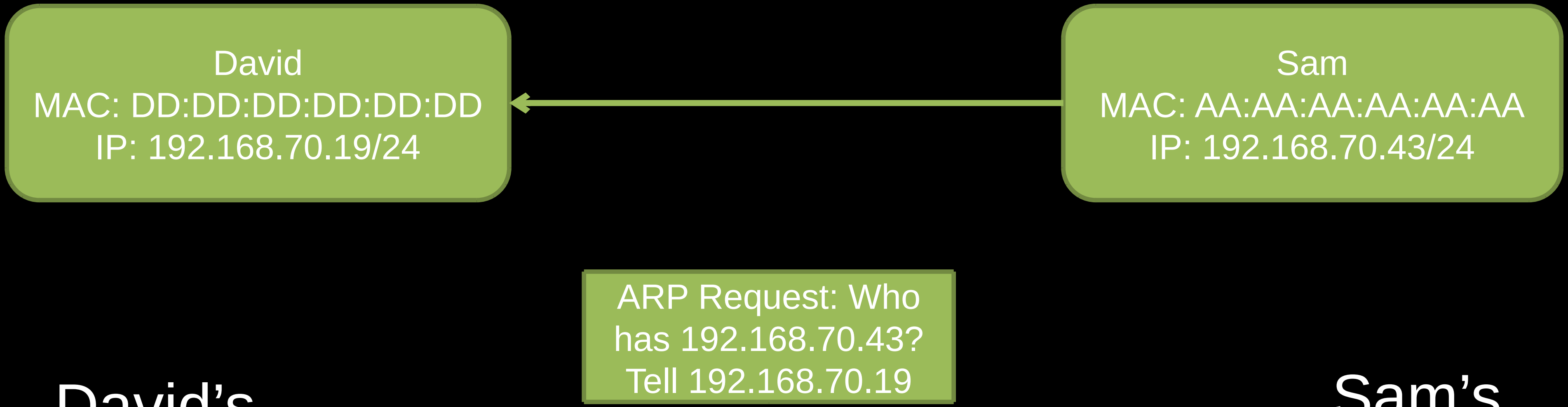
David's ARP Table

IP Address	MAC Address

Sam's ARP Table

IP Address	MAC Address

ARP Example



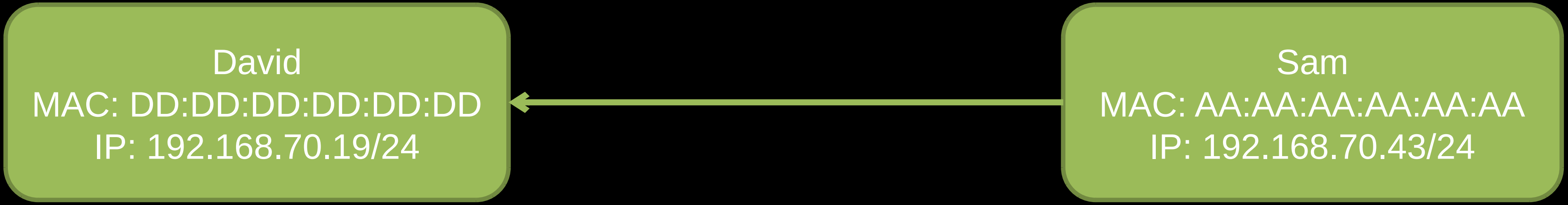
David's ARP Table

IP Address	MAC Address

Sam's ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Example



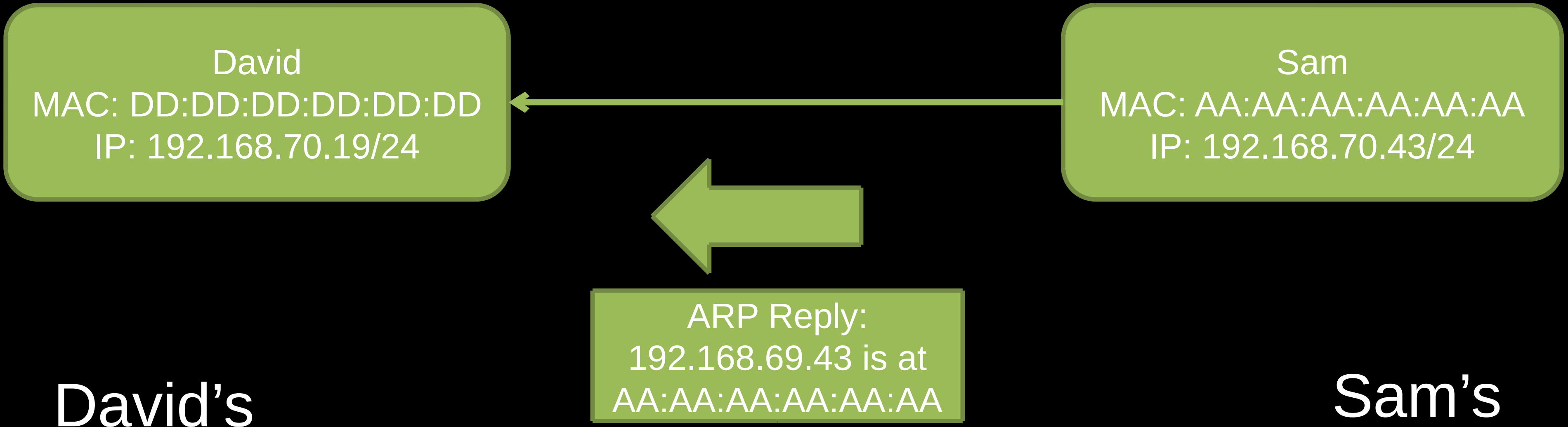
David's
ARP Table

IP Address	MAC Address

Sam's
ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Example



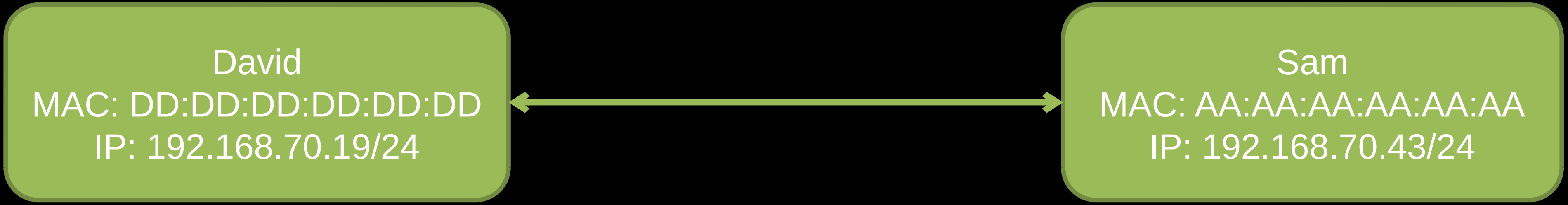
David's ARP Table

IP Address	MAC Address

Sam's ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Example



David's ARP Table

IP Address	MAC Address
192.168.70.43	AA:AA:AA:AA:AA:AA

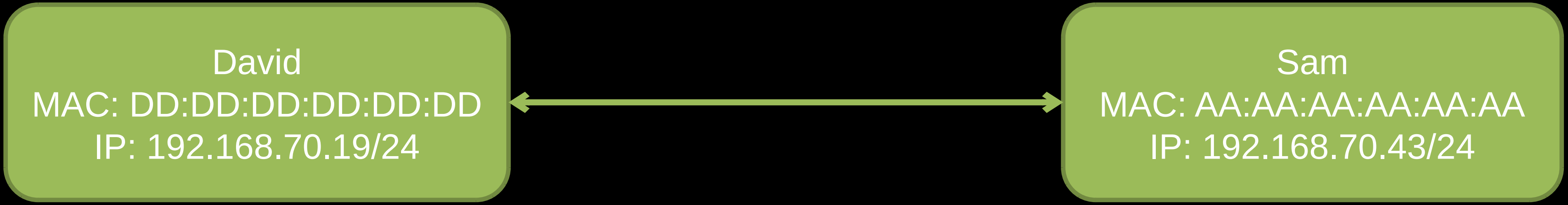
Sam's ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing

- Devices often store data from ARP replies in a table or cache to avoid sending multiple ARP requests for the same IP address
- The ARP cache stores any data from observed ARP replies, even if the device made no associated request
- If a target gets flooded with gratuitous ARP replies with falsified information, the target trusts that information and updates its ARP cache
- This technique allows one to impersonate an IP address to a target device
- Such impersonation can achieve a “Man in the Middle” and enable one to observe or manipulate network traffic
- Limitation: all targets and the impersonator must be on the same subnet

ARP Spoofing Example



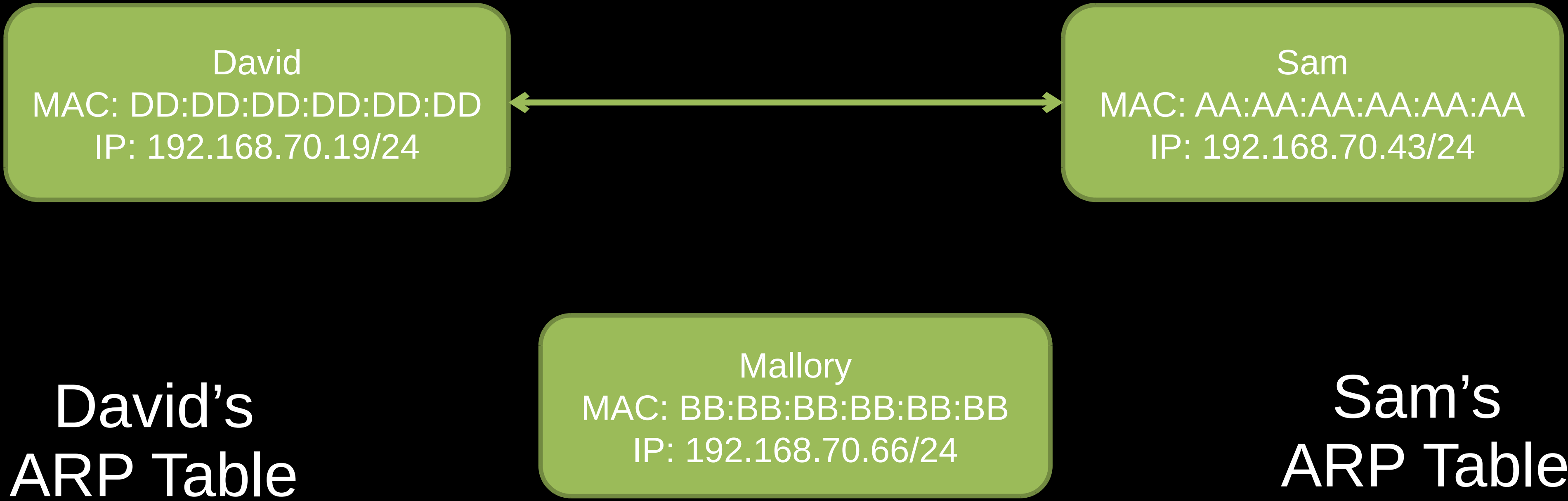
David's
ARP Table

IP Address	MAC Address
192.168.70.43	AA:AA:AA:AA:AA:AA

Sam's
ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing Example



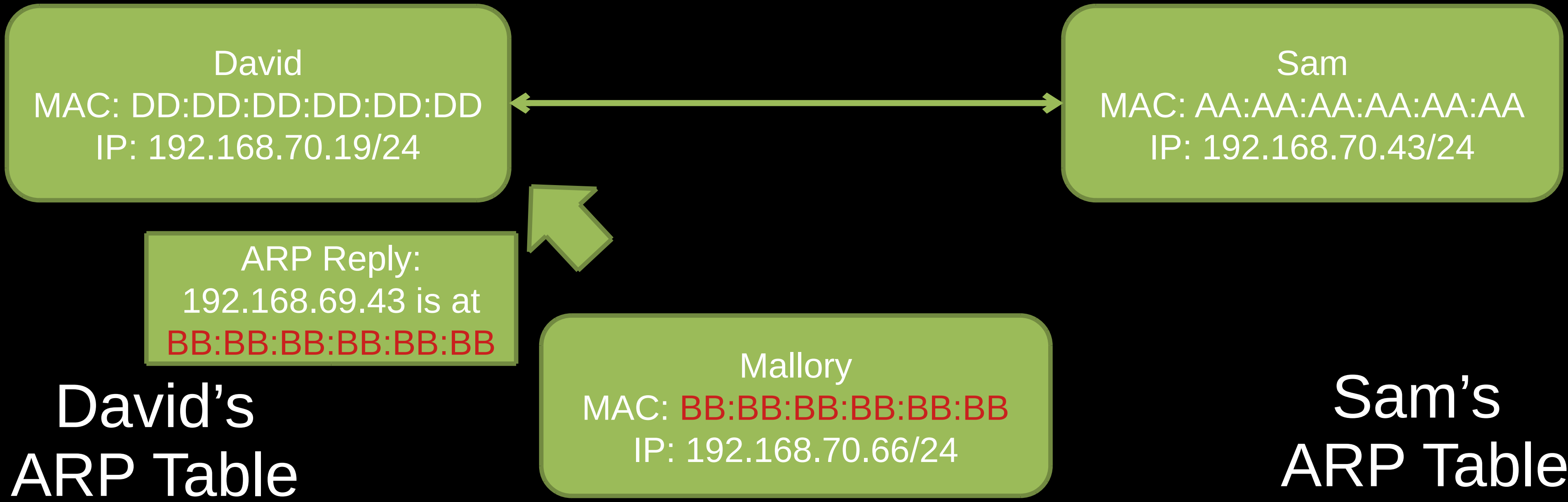
David's
ARP Table

IP Address	MAC Address
192.168.70.43	AA:AA:AA:AA:AA:AA

Sam's
ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing Example



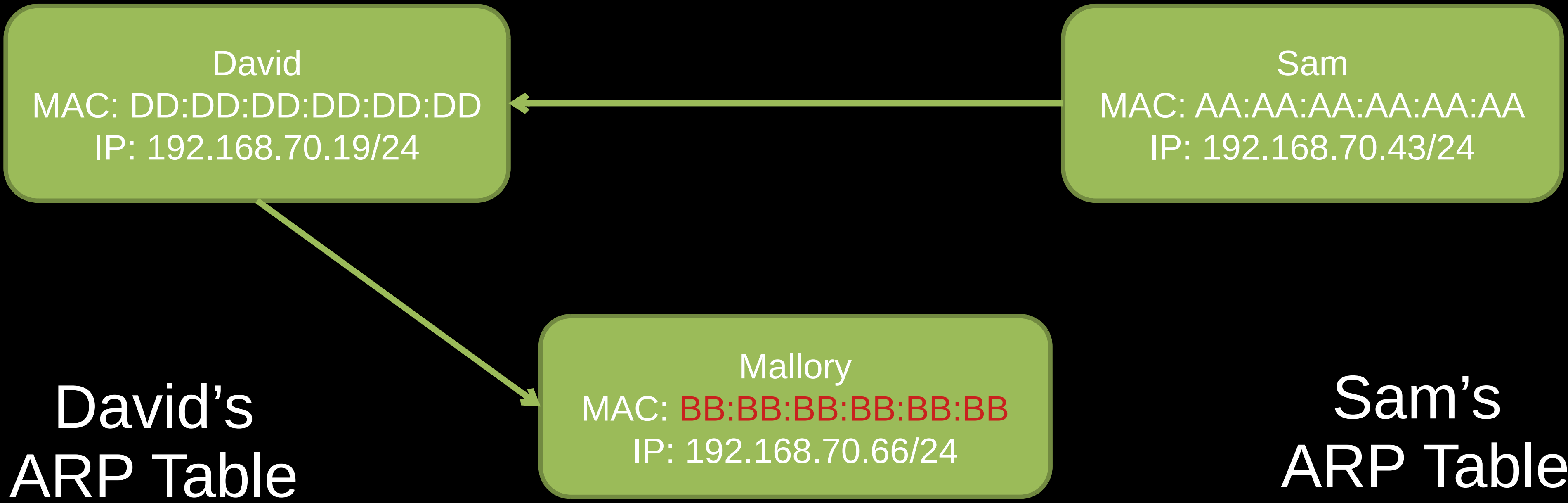
David's ARP Table

IP Address	MAC Address
192.168.70.43	AA:AA:AA:AA:AA:AA

Sam's ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing Example



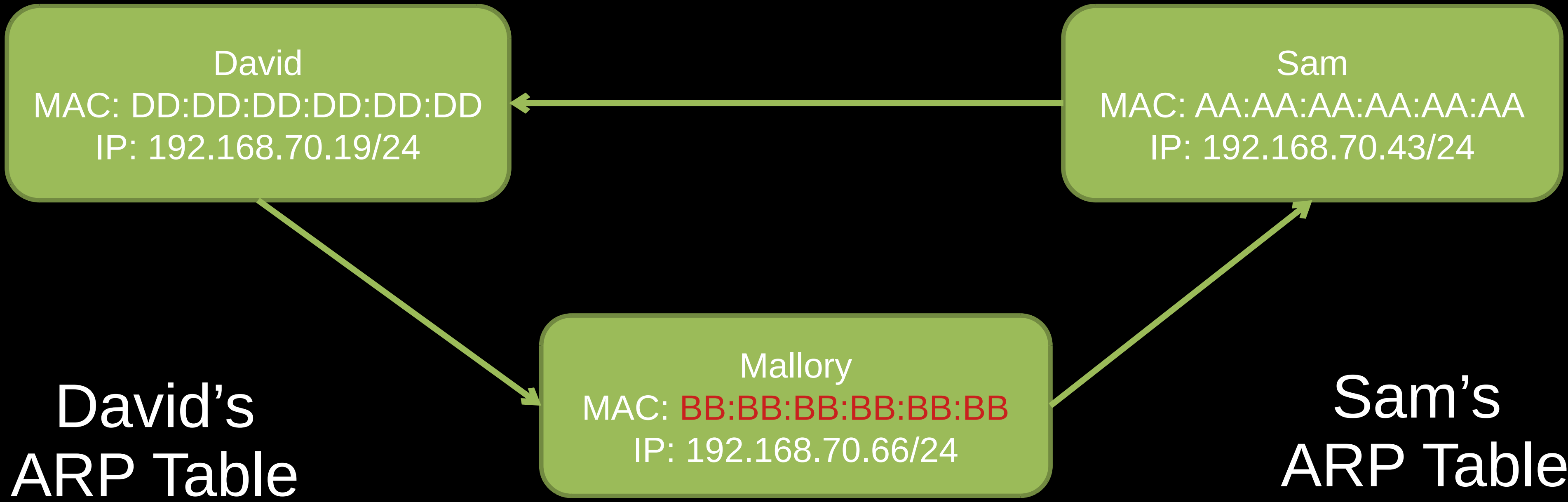
David's
ARP Table

IP Address	MAC Address
192.168.70.43	BB:BB:BB:BB:BB:BB

Sam's
ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing Example



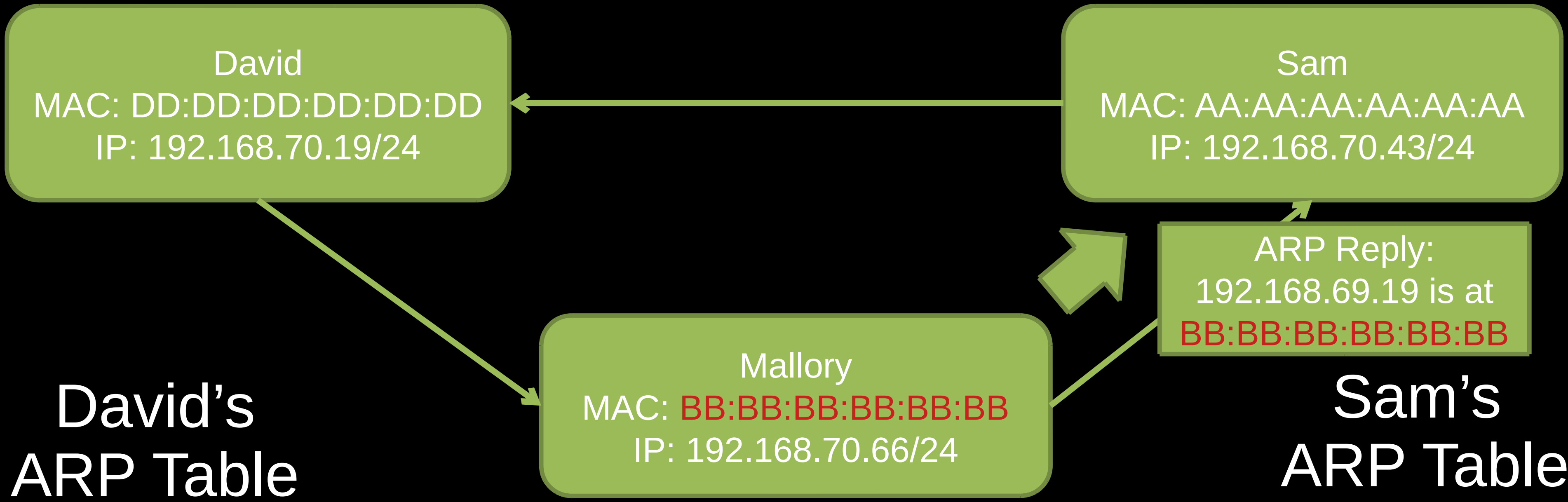
David's
ARP Table

IP Address	MAC Address
192.168.70.43	BB:BB:BB:BB:BB:BB

Sam's
ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing Example



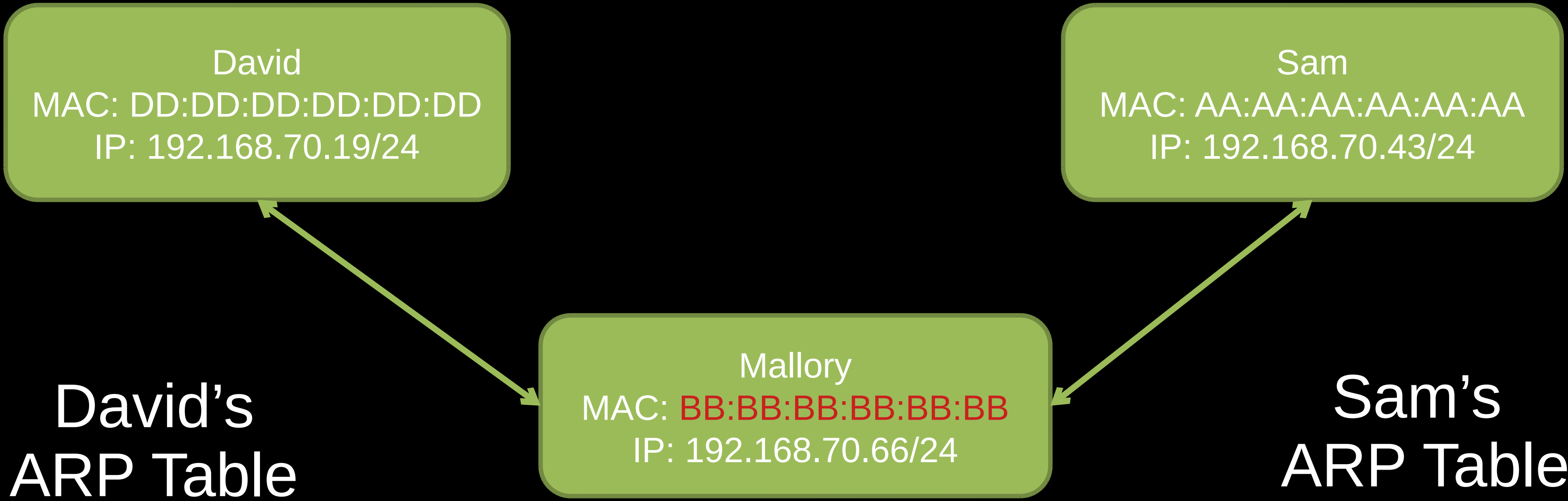
David's
ARP Table

IP Address	MAC Address
192.168.70.43	BB:BB:BB:BB:BB:BB

Sam's
ARP Table

IP Address	MAC Address
192.168.70.19	DD:DD:DD:DD:DD:DD

ARP Spoofing Example



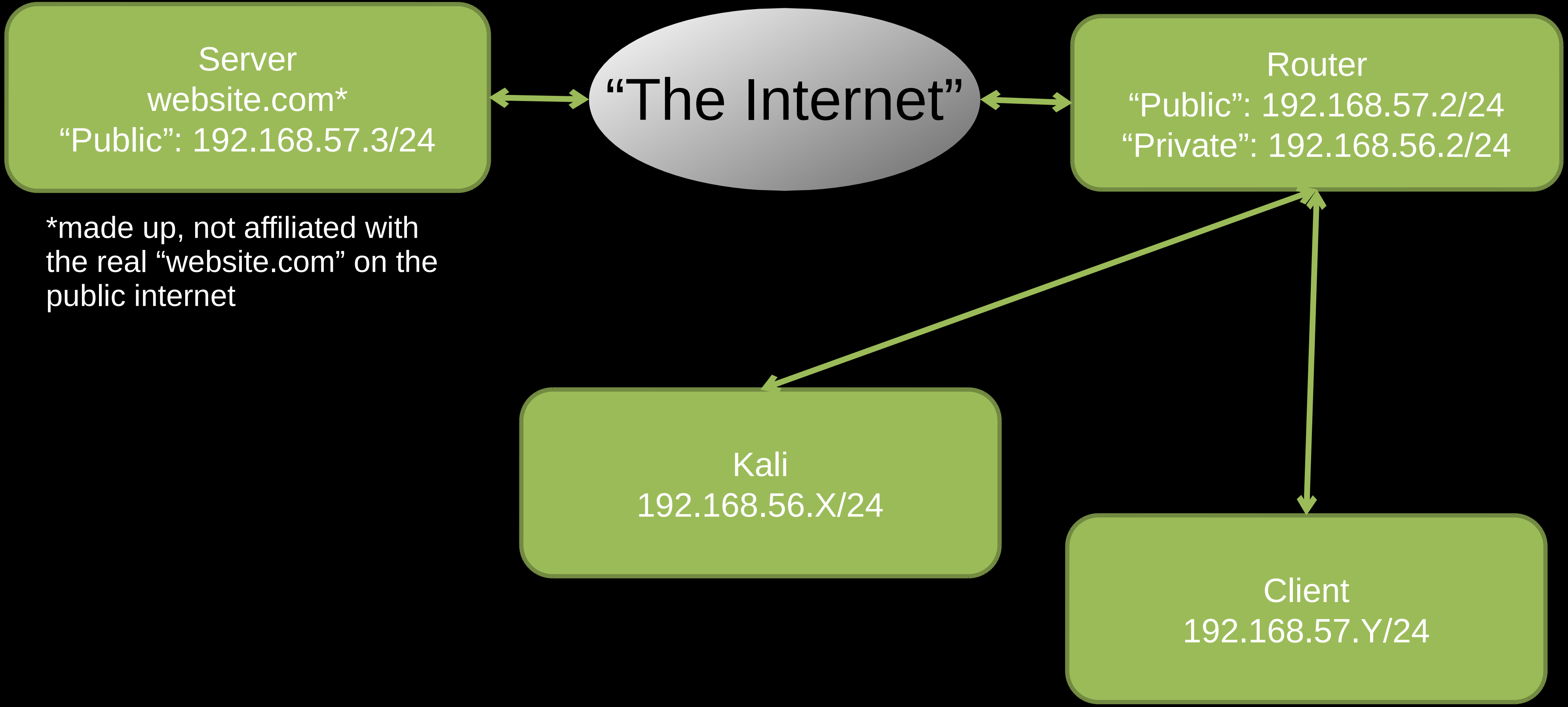
David's ARP Table

IP Address	MAC Address
192.168.70.43	BB:BB:BB:BB:BB:BB

Sam's ARP Table

IP Address	MAC Address
192.168.70.19	BB:BB:BB:BB:BB:BB

ARP Spoofing Demo



*made up, not affiliated with the real “website.com” on the public internet

Dynamic Host Configuration Protocol (DHCP)

- DHCP performs automatic configuration of network settings when a device connects to a network
- The client broadcasts a message to ask if any DHCP server exists on the network
- The DHCP server replies with a DHCP lease with an IP address, subnet mask, and other optional settings such as the DNS server and default gateway
- The client must renew the lease periodically

Dynamic Host Configuration Protocol (DHCP)

- DHCP message types:
  - DHCP Discovery: broadcast to ask if any DHCP server exists
  - DHCP Offer: DHCP server offers a DHCP lease to the client who sent the DHCP Discover
  - DHCP Request: the client tells the server it wants to accept the lease
  - DHCP Acknowledge: the server confirms the client can use the information in the lease

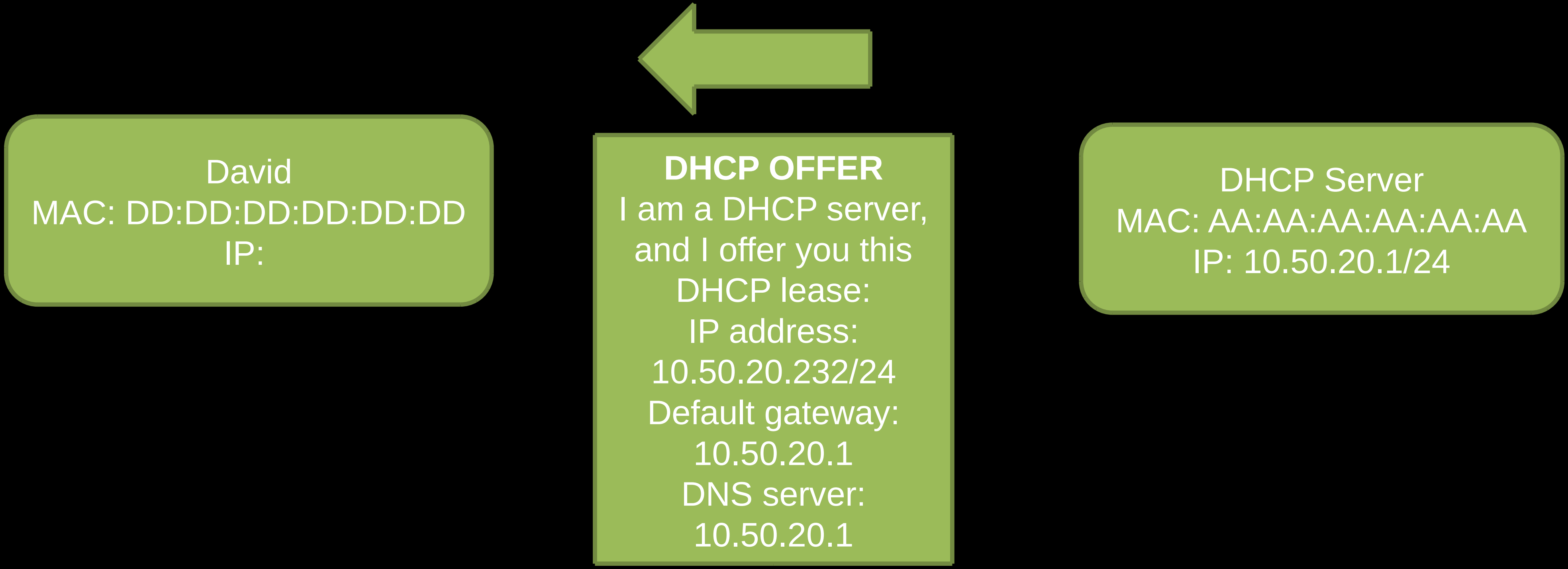
DHCP Example

DHCP DISCOVERY
Does a DHCP server exist?

David
MAC: DD:DD:DD:DD:DD:DD
IP:

DHCP Server
MAC: AA:AA:AA:AA:AA:AA
IP: 10.50.20.1/24

DHCP Example

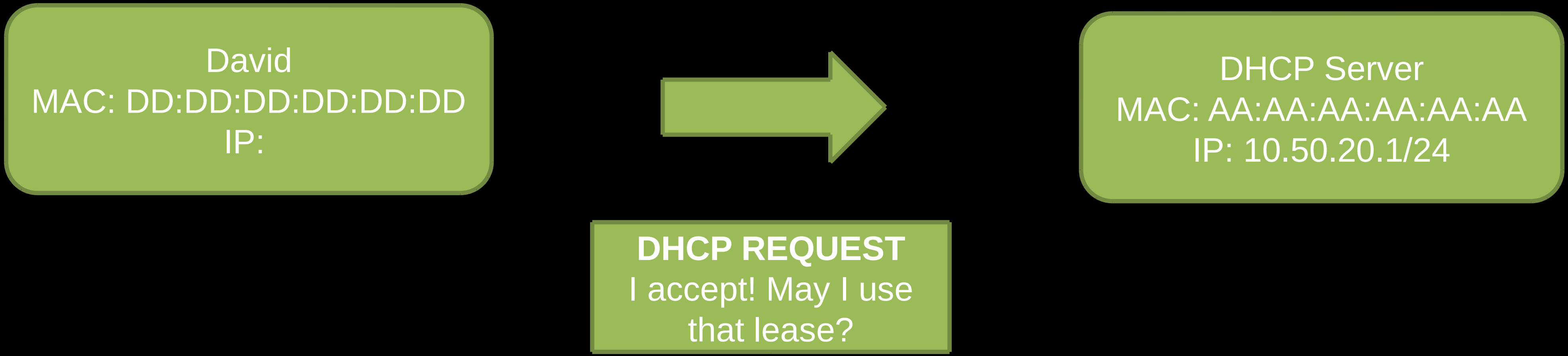


David
MAC: DD:DD:DD:DD:DD:DD
IP:

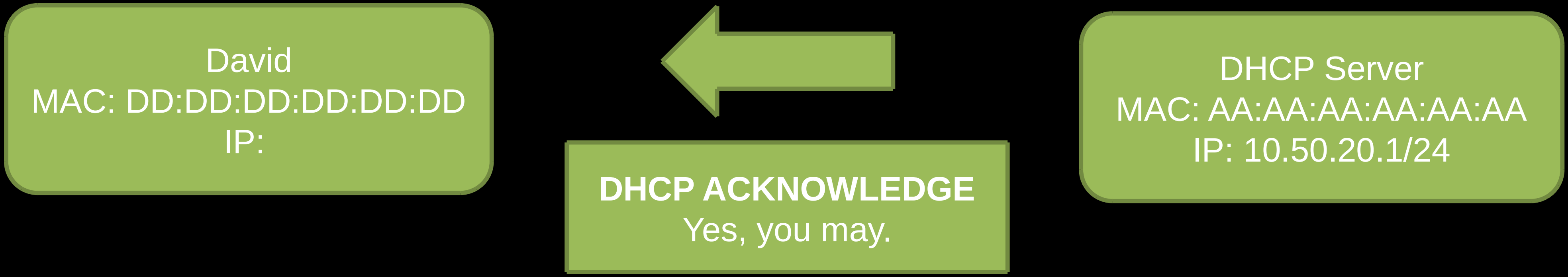
DHCP OFFER
I am a DHCP server, and I offer you this DHCP lease:
IP address: 10.50.20.232/24
Default gateway: 10.50.20.1
DNS server: 10.50.20.1

DHCP Server
MAC: AA:AA:AA:AA:AA:AA
IP: 10.50.20.1/24

DHCP Example



DHCP Example



DHCP Example

David
MAC: DD:DD:DD:DD:DD:DD
IP: 10.50.20.232/24

DHCP Server
MAC: AA:AA:AA:AA:AA:AA
IP: 10.50.20.1/24

DHCP Spoofing: Denial of Service

- Resource starvation attack
- Flood the target server with DHCP Discover messages with falsified client MAC addresses
- The target server makes offers until it runs out of IP addresses
- New clients time out when they do not receive a DHCP Offer from the target server

Rogue DHCP Server

- Set up a rogue DHCP server with desired options for settings such as DNS server, default gateway, etc.
- Whenever the rogue DHCP server sends an offer faster than the legitimate DHCP server, the client uses the settings offered by the rogue server
- To remove this race condition, perform a DHCP starvation attack on the legitimate server
- This technique grants control over any network setting controlled by DHCP on the target device

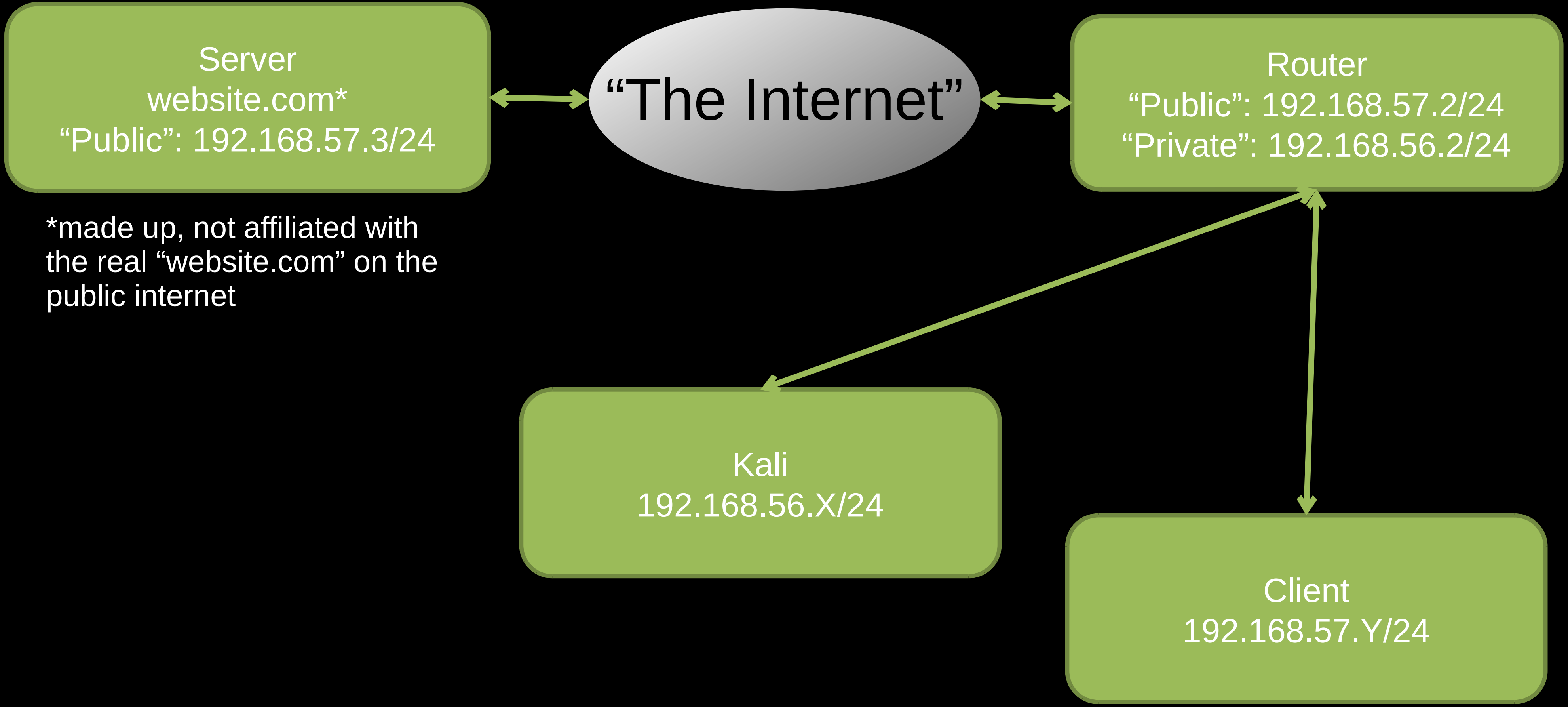
Domain Name Service (DNS)

- DNS associates a human-friendly domain name such as www.example.com with an IP address
- The client sends a DNS query to its configured DNS server and waits for a response
- The DNS server replies with the IP address for the queried domain name or an empty answer
- DNS queries use UDP port 53

DNS Spoofing

- A man in the middle can observe DNS queries and responses
- Manipulation or forgery of the response enables impersonation of a domain name to the device that sent the query
  - Several options:
    - Craft and send a spoofed DNS reply and win a race with the legitimate reply
    - Craft and send a spoofed reply and do not forward the request to the DNS server
    - Modify the legitimate response before forwarding it to the device that sent the query
- Depending on the target device, the information in the spoofed reply may get stored in a DNS cache and used until it expires

DNS Spoofing Demo



*made up, not affiliated with the real "website.com" on the public internet

Network Time Protocol (NTP)

- Synchronizes clocks over a network
- Usually achieves synchronization within tens of milliseconds over the internet
- Can achieve <1ms synchronization accuracy over local networks
- An NTP client:
  - Polls one or more NTP servers at regular intervals
  - Calculates round-trip delay
  - Uses the calculated round-trip delay to calculate the offset of the local clock from the NTP server clock
  - Updates the local clock
- NTP uses UDP port 123

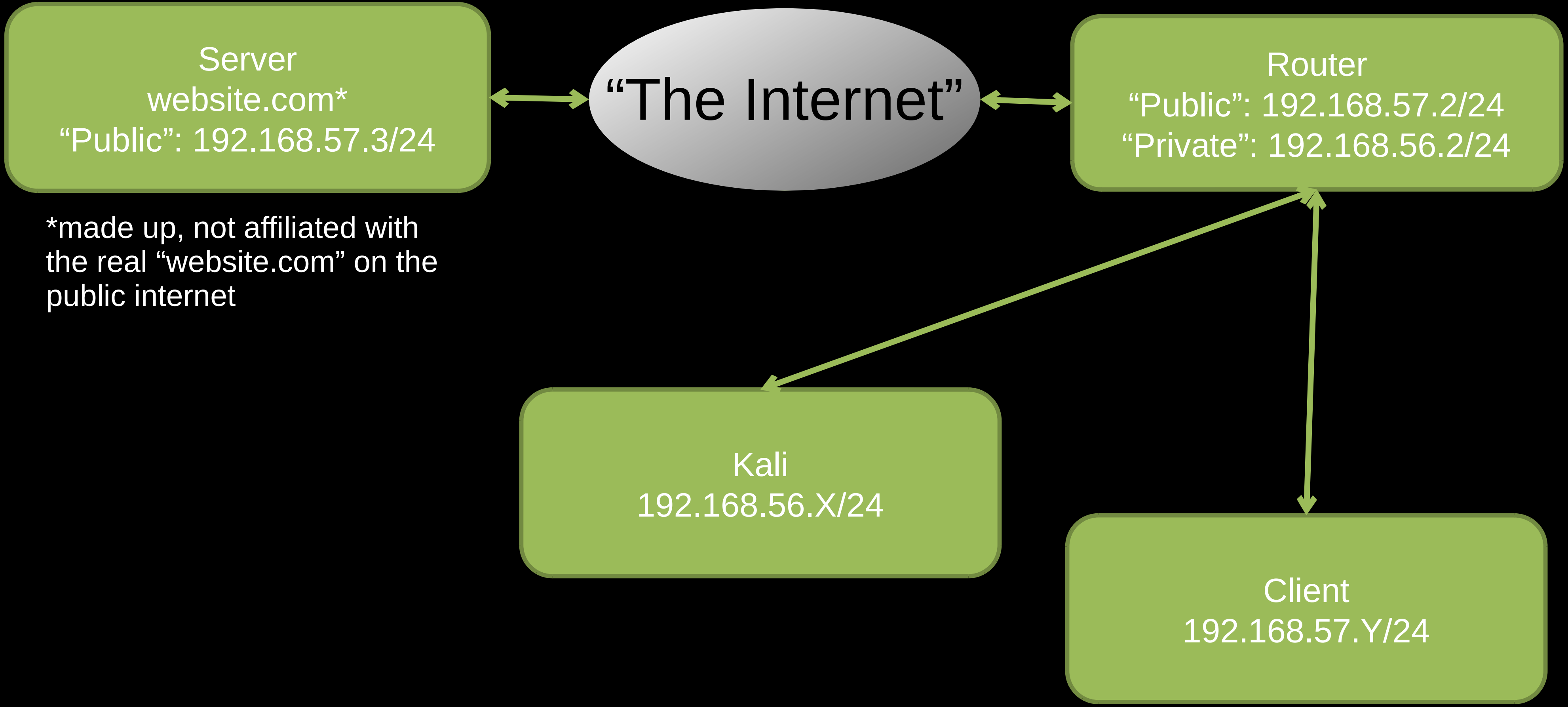
NTP Spoofing

- A man in the middle can manipulate or forge NTP query responses
- The NTP client trusts the falsified reply and sets the system clock accordingly
- The man in the middle may update the clock on the target device whenever the target performs an NTP query

NTP Spoofing

- Potential impacts of changing system clock:
  - Interfere with scheduled/automated tasks
  - Expire cached items such as DNS cache entries
  - Disrupt Kerberos (or any timing-sensitive) authentication
  - Force expiration of cryptographic keys or certificates

NTP Spoofing Demo



*made up, not affiliated with the real “website.com” on the public internet

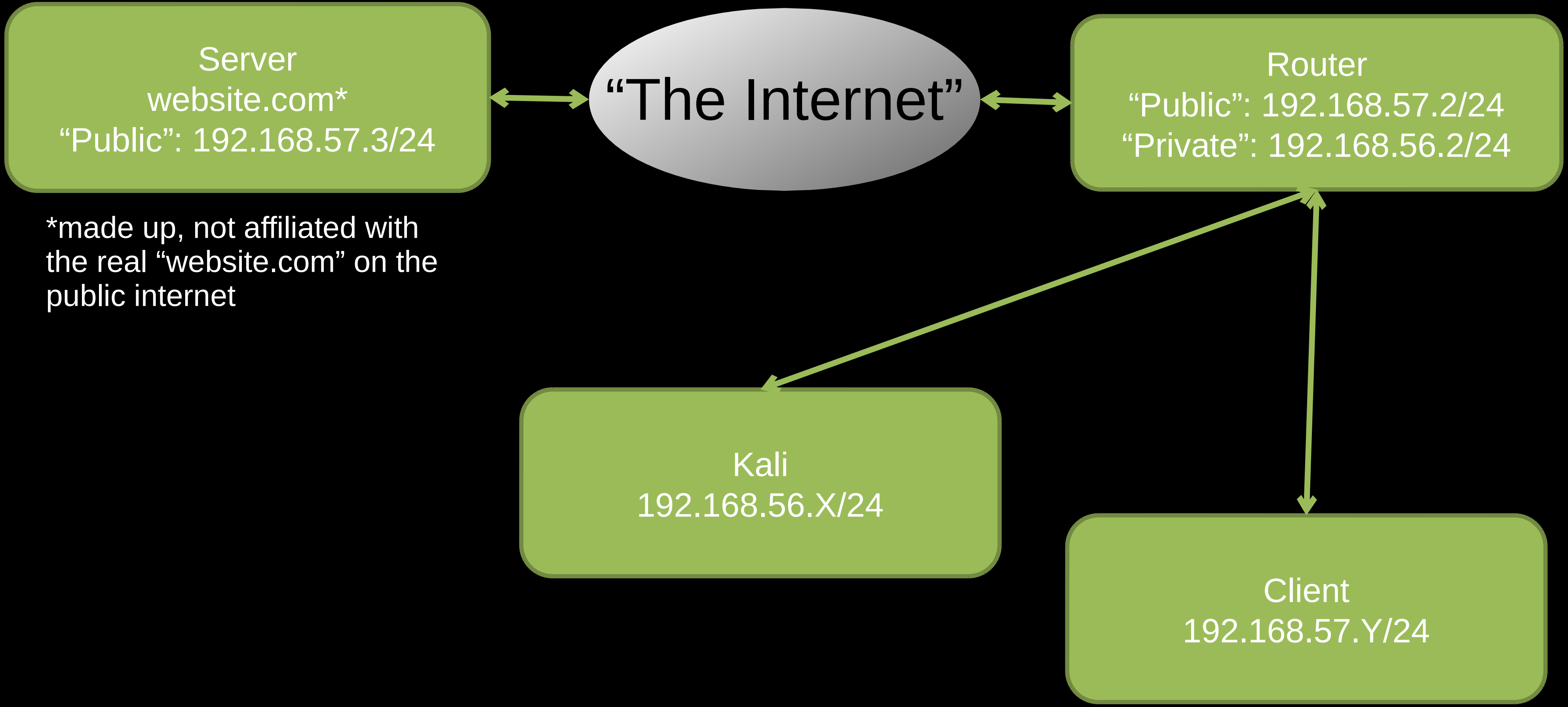
Hypertext Transfer Protocol (HTTP)

- HTTP is the protocol a web browser uses to browse the web
- Client-server protocol: a client sends a **request**, and the server replies with a **response**
- Requests use a **method**
  - **GET**: ask the web server to respond with a resource
  - **POST**: submit information such as a username and password
- Allows clients to request resources such as web pages or images, or submit data such as form entries

HTTP Security Concerns

- HTTP includes no mechanisms for authentication or confidentiality of data
- A man in the middle can:
  - Observe data in HTTP requests or responses, including login credentials
  - Add, change, or remove content of web pages by manipulating response data
  - Add arbitrary JavaScript to web pages
  - Modify parameters the client sends in POST requests

HTTP Eavesdropping Demo



*made up, not affiliated with the real “website.com” on the public internet

Foundational Topics

- Encryption
- Authentication
- Diffie-Hellman Key Exchange
- Public Key Infrastructure

Encryption

- Protects the confidentiality of data
- Uses a key to turn plaintext into ciphertext or ciphertext into plaintext
- Ciphertext ideally looks indistinguishable from random data
- Symmetric: the same key encrypts and decrypts data
- Asymmetric: a public key encrypts, a private key decrypts

Encryption does not (always) imply authentication!

Authentication

- Under certain assumptions:
  - Verifies the identity of an entity
  - Verifies a message came from its stated source
  - Implies the verification of integrity
- Often implemented with features of cryptography
  - Cryptographic signatures
  - Hash-based message authentication codes (HMAC)

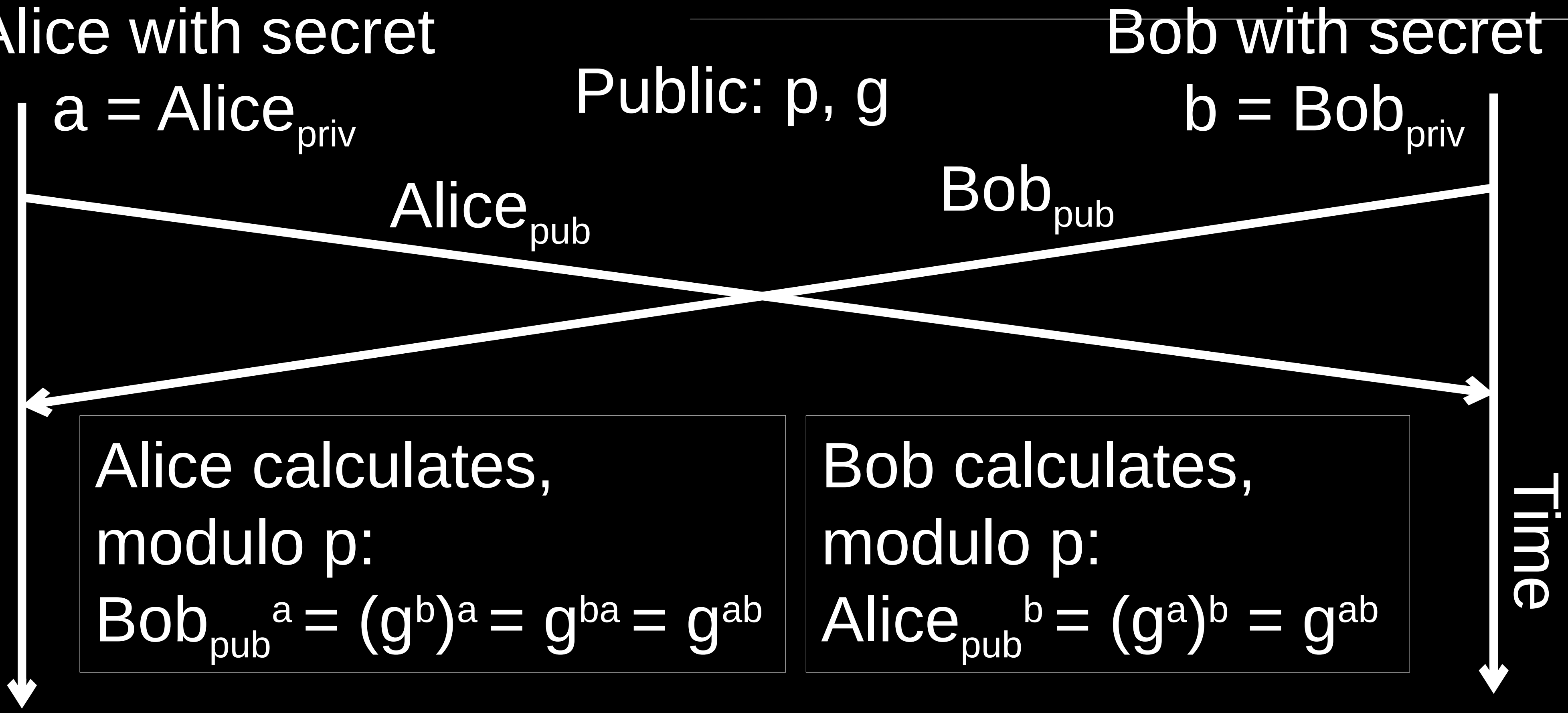
Diffie-Hellman Key Exchange

- Process for two parties, Alice and Bob, to agree on a shared secret without a secure channel
- Relies on the computational difficulty of the Discrete Logarithm Problem

Diffie-Hellman Key Exchange

- Private information: a secret for each of the two parties
  - a or $Alice_{priv}$
  - b or Bob_{priv}
- Public information:
  - p: a large prime number
  - g: a generator of the group Z_p
  - $Alice_{pub} = g^a \bmod p$
  - $Bob_{pub} = g^b \bmod p$

Diffie-Hellman Key Exchange


Alice with secret
$a = \text{Alice}_{\text{priv}}$
Public: p, g
Bob with secret
$b = \text{Bob}_{\text{priv}}$
$\text{Alice}_{\text{pub}}$
Bob_{pub}
Time

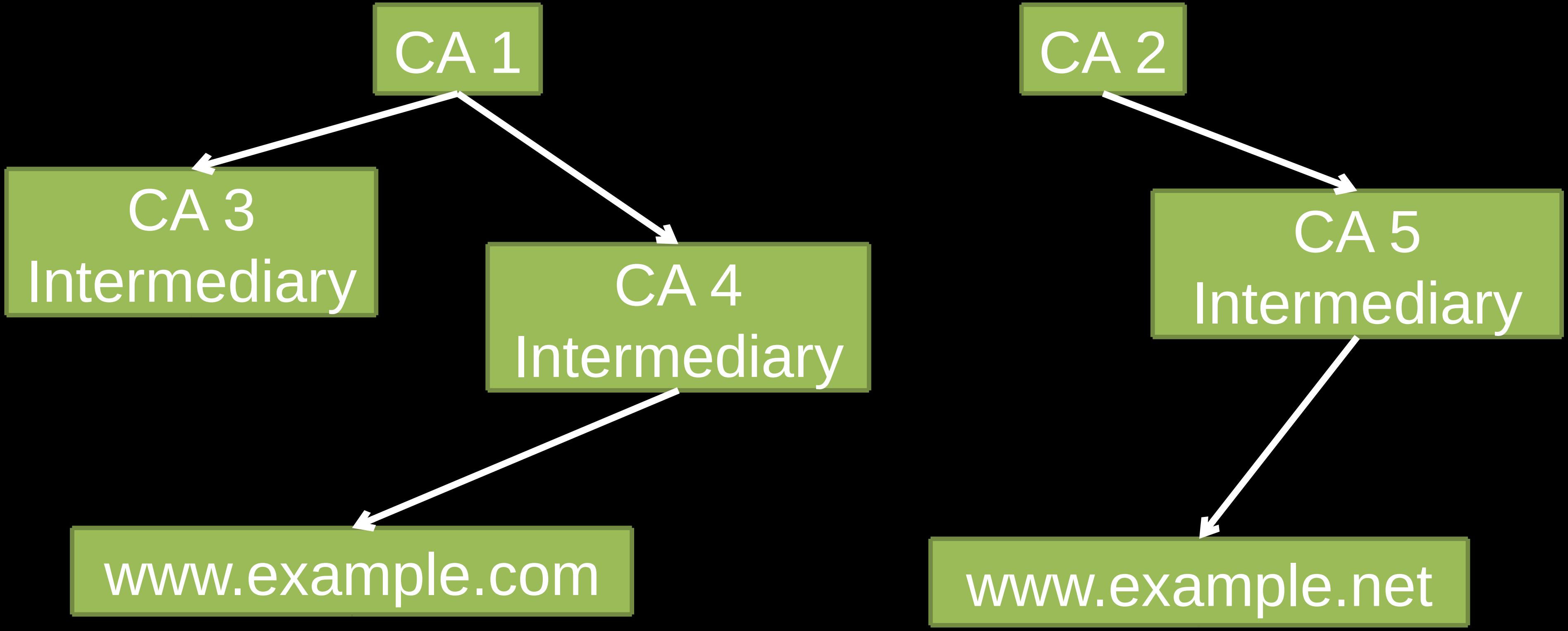
Alice calculates, modulo p:
$\text{Bob}_{\text{pub}}^{a} = (g^{b})^{a} = g^{ba} = g^{ab}$

Bob calculates, modulo p:
$\text{Alice}_{\text{pub}}^{b} = (g^{a})^{b} = g^{ab}$

Public Key Infrastructure (PKI)

- All asymmetric cryptography faces the issue of associating a public key with the correct identity
- With PKI, a trusted third party attests to the ownership and validity of a public key
- Devices trust Certificate Authorities (CAs) as the trusted third parties
- CAs may sign certificates granting other, intermediary CAs the authority to sign certificates

Public Key Infrastructure (PKI)



Transport Layer Security (TLS)

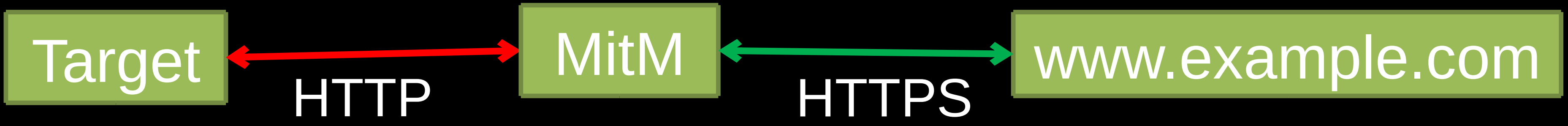
- Designed to add security to any higher-level protocol
- Protects confidentiality with encryption
- Supports optional authentication for one or both parties
- Authentication uses public key infrastructure
- Handshake determines the latest TLS version and strongest cipher suite supported by both parties
- Supports Diffie-Hellman Key Exchange for key agreement for symmetrical encryption and forward secrecy
- Successor of Secure Sockets Layer (SSL)

HTTP Secure (HTTPS)

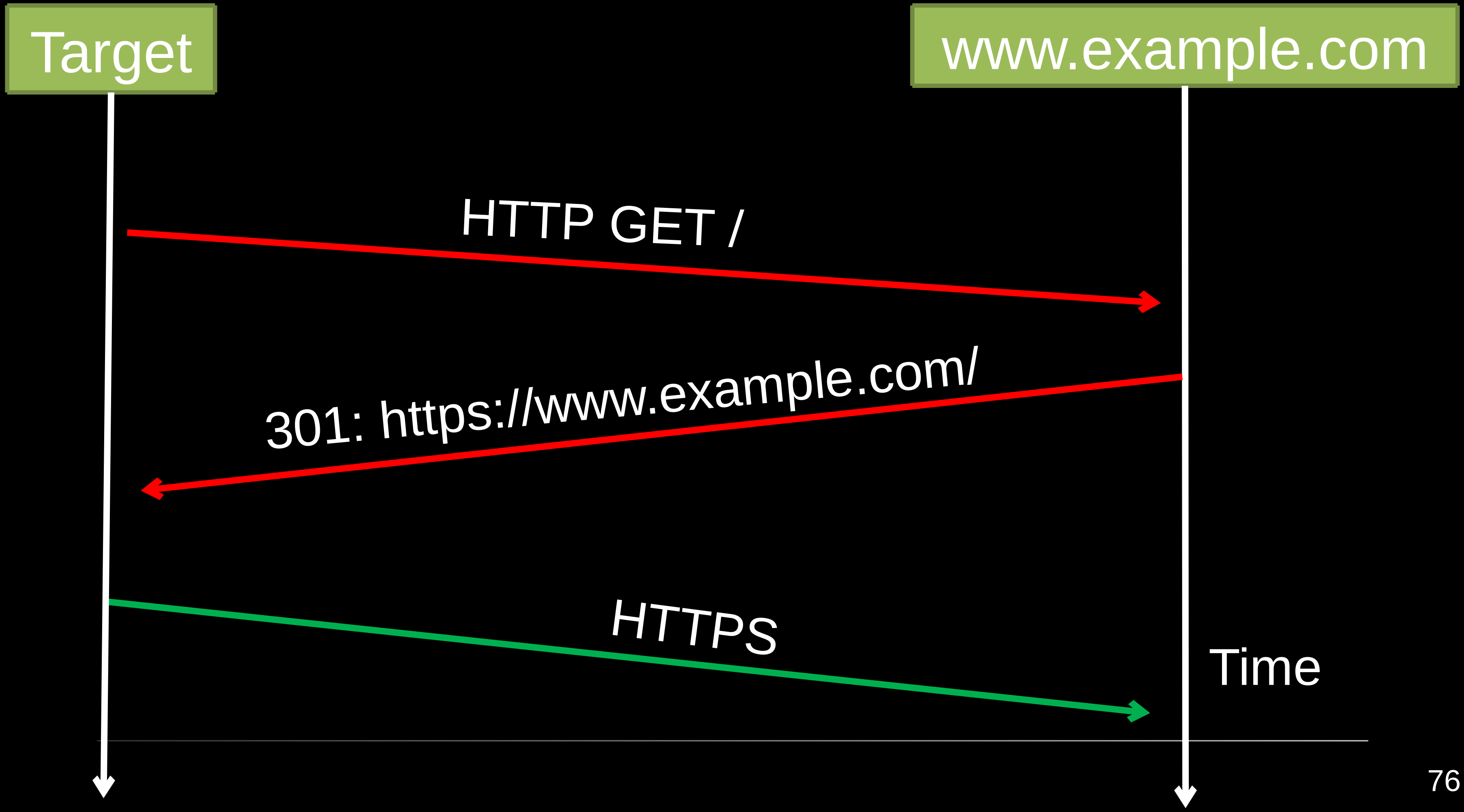
- HTTPS addresses the security concerns of HTTP by encapsulating HTTP traffic in TLS
- TLS provides confidentiality and server authentication
- Websites present a signed certificate with the domain name and public key of the web server
- The web server authenticates its public Diffie-Hellman value when negotiating with the client
- The client often does not authenticate itself or its Diffie-Hellman value
  - Is that problematic?

SSL Stripping

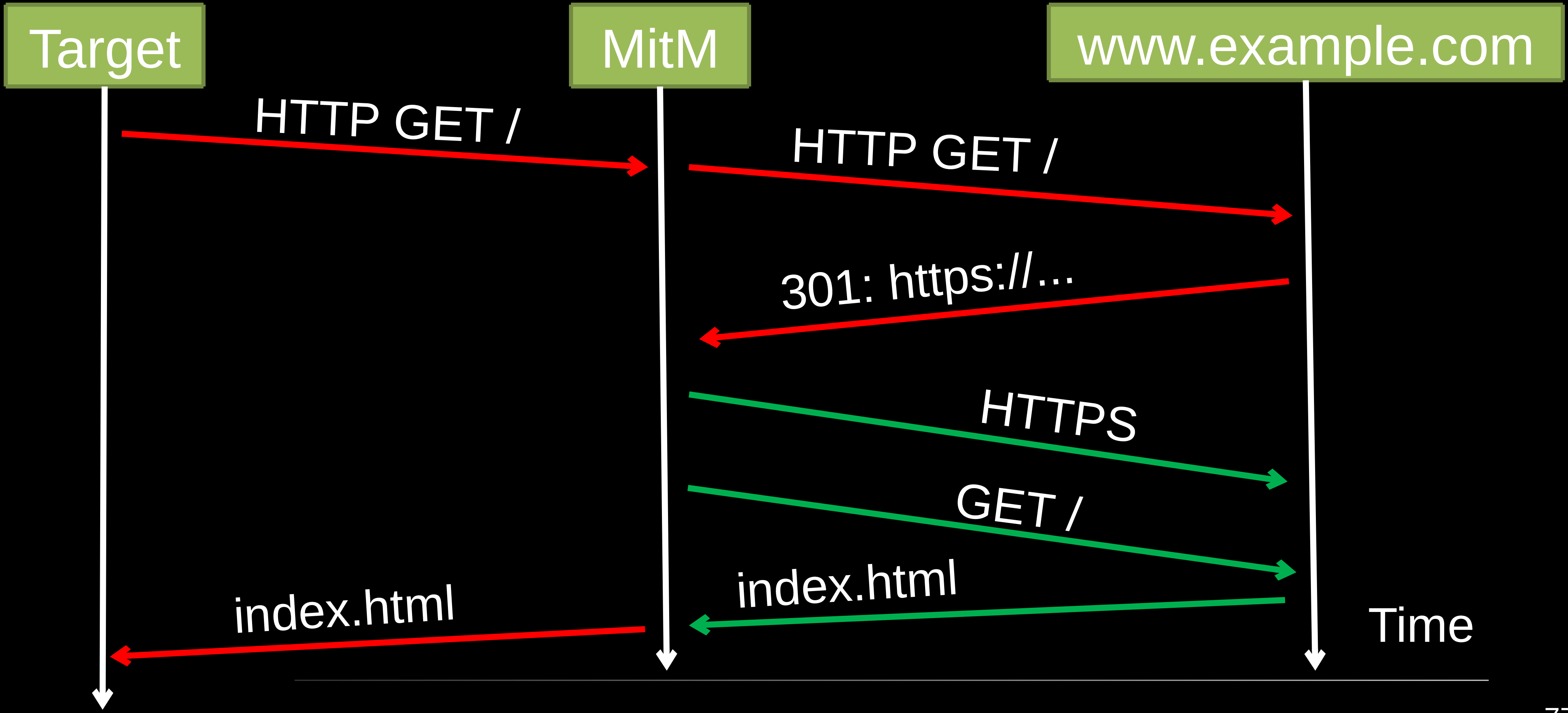
- Assumes the target browser connects to a website with HTTP by default
- Web servers conventionally reply with an HTTP 301 redirect to HTTPS
- SSL stripping uses a man in the middle attack to connect to the web server over HTTPS but keeps the client communication with the man in the middle over HTTP



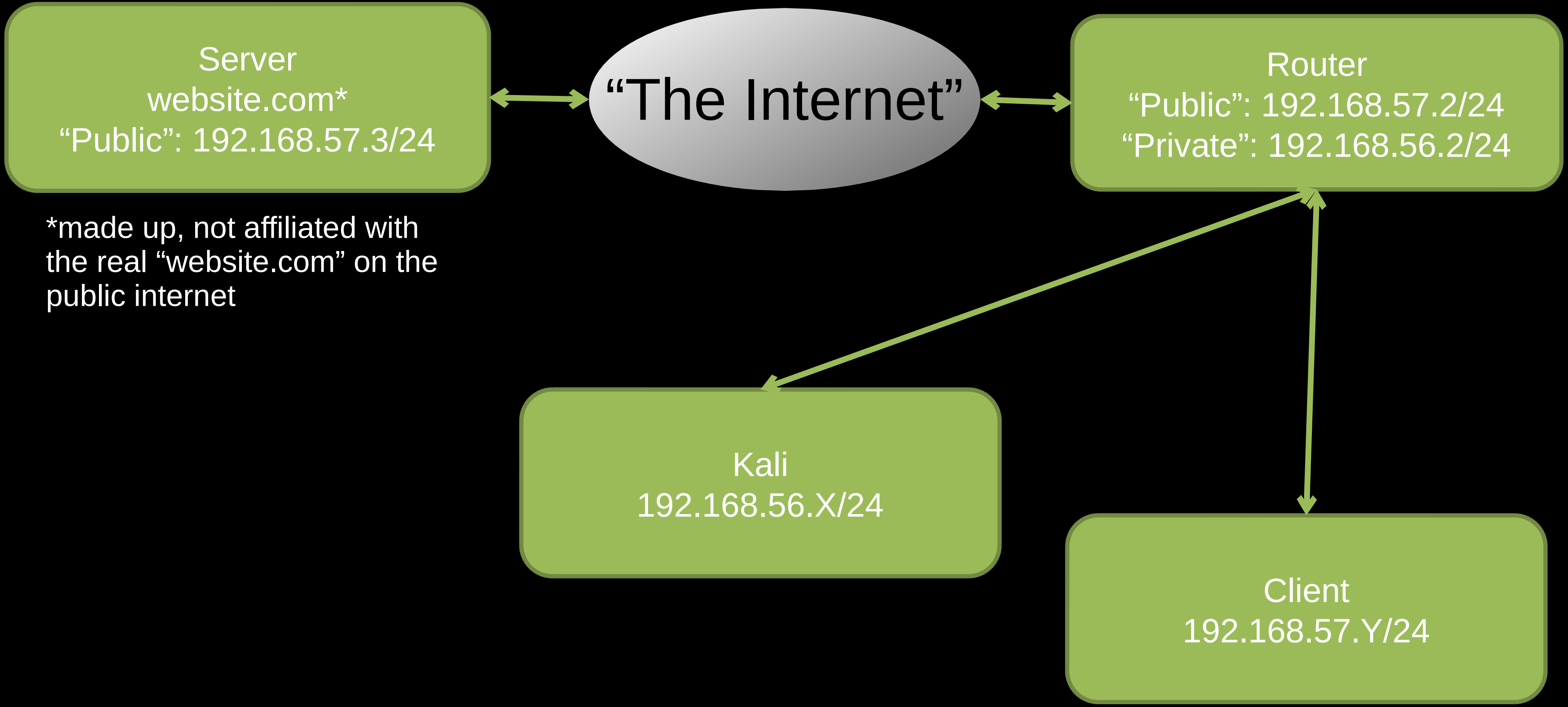
Without SSL Stripping



With SSL Stripping



SSL Stripping Demo

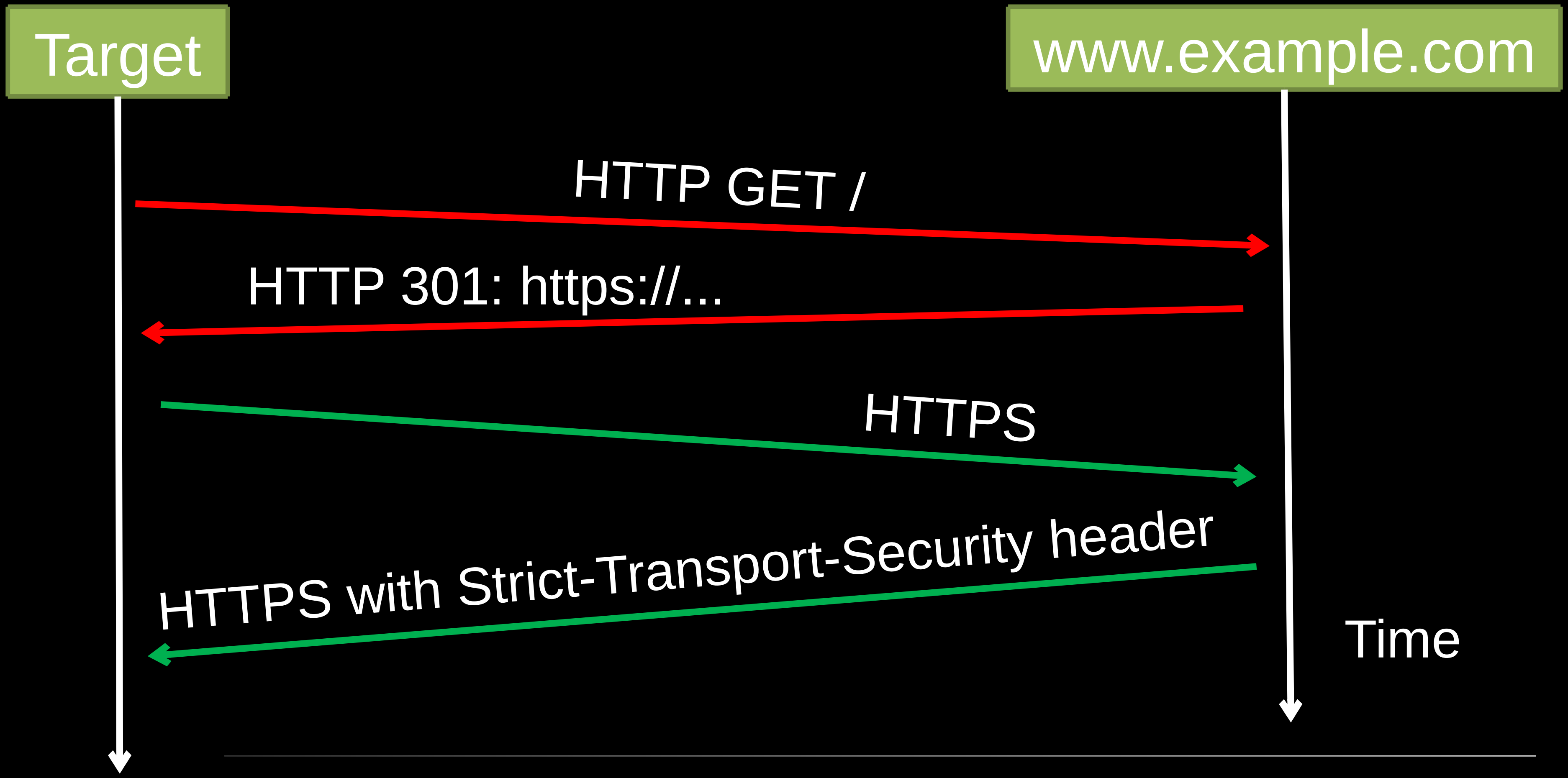


*made up, not affiliated with the real “website.com” on the public internet

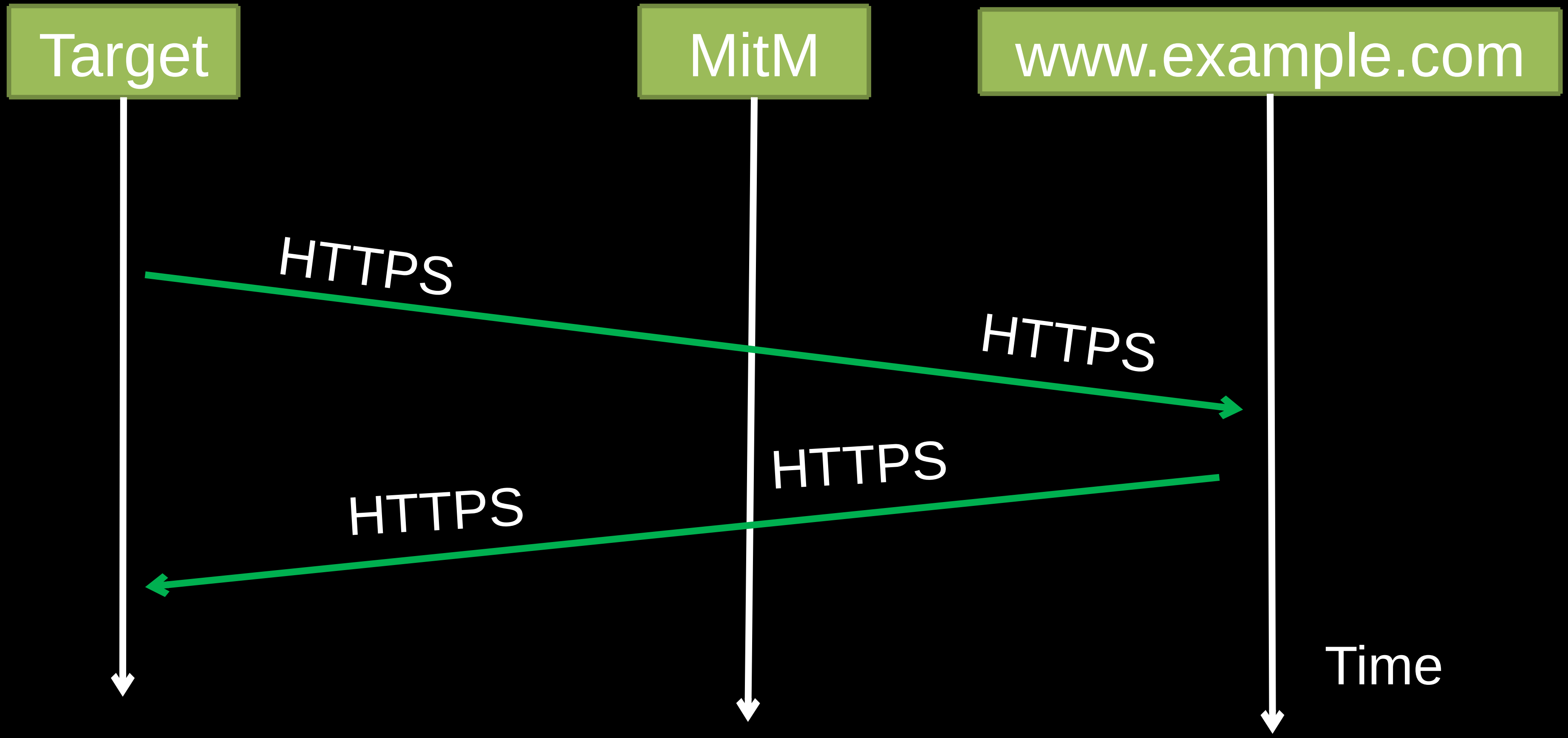
HTTP Strict Transport Security (HSTS)

- Assumption: a user visits a website for the first time with no malicious actor in play
- The web server sends HTTPS responses with the HSTS header set
- The HSTS header tells the web browser to only contact the web server over HTTPS until a stated expiration date, usually around one year in the future
- HSTS protects against protocol downgrade attacks similar to SSL stripping

With HSTS: First Time Connecting



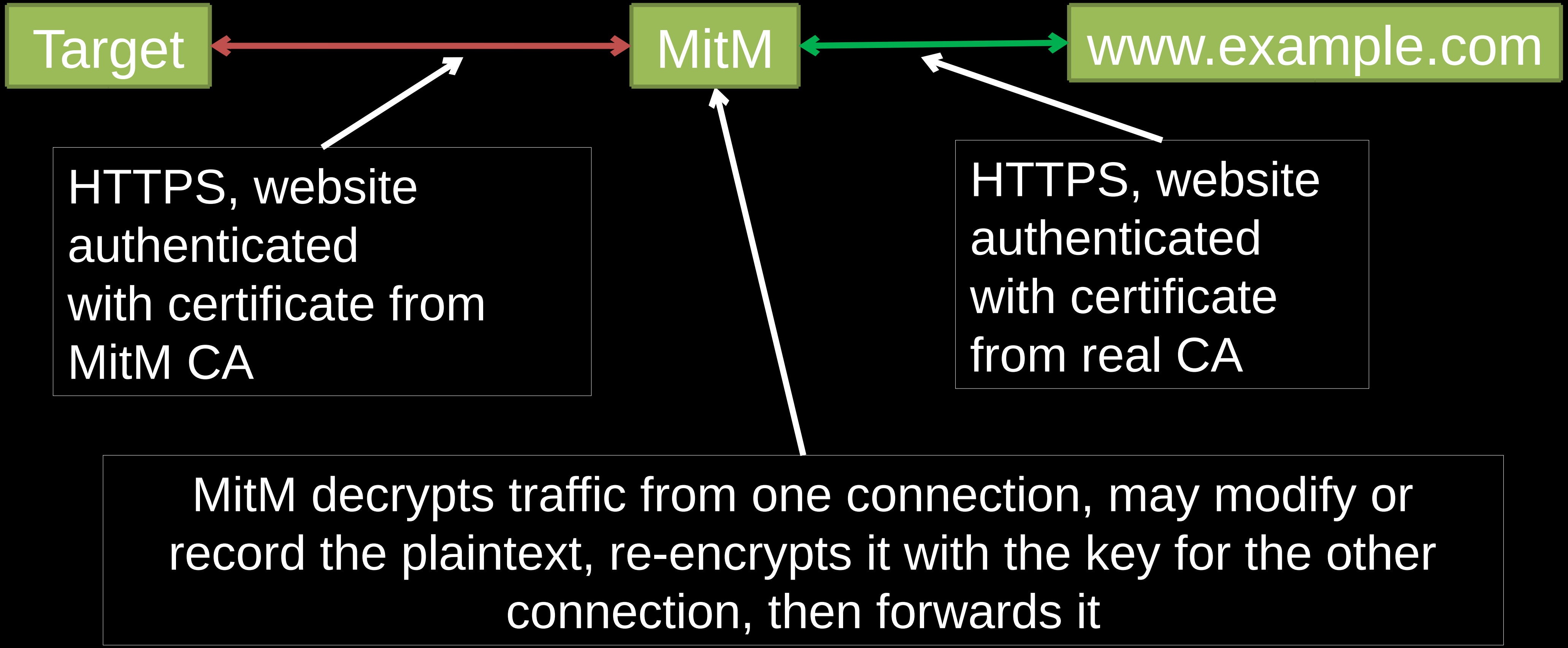
With HSTS: Subsequent Connections



SSL/TLS Splitting

- Two primary requirements:
  - Control over a CA trusted on the target device
  - A man in the middle
- When the target tries to visit a site over HTTPS, impersonate the website with a certificate generated and signed on the fly by the trusted CA
- Connect to the real website on behalf of the target device
- Forward data between the target and website, decrypting and re-encrypting it between the two HTTPS connections

SSL/TLS Splitting



HTTPS, website authenticated with certificate from MitM CA

HTTPS, website authenticated with certificate from real CA

MitM decrypts traffic from one connection, may modify or record the plaintext, re-encrypts it with the key for the other connection, then forwards it

Overview

- Networking and OSI model review
- Network protocols, trust models, and attacks

- ARP	- DNS	- TLS
- DHCP	- NTP	- HTTP(S)

- **Discussion**

Discussion Topics

- Do TLS and HTTPS solve all the problems?
- How does the trust model of SSH work? How is it different from the protocols we discussed today?
- Should every layer of the OSI model support security features such as authentication?