IERG 4130 – 2026 Fall

香港中文大學
CUHK

Low-Level Network Attacks

Yajin ZHOU

https://yajin.org

Credits: the slides are from CS161 of UC Berkeley

Low-Level Network Attacks



- **Network Attackers**
  - Man-in-the-middle attacker
  - On-path attacker
  - Off-path attacker
- **ARP: Translate IP addresses to MAC addresses**
- **WPA: Communicate securely in a wireless local network**
- **DHCP: Get configurations when first connecting to a network**

Types of Network Attackers

- **Threat model: There are 3 types of attackers we'll consider**

	Can modify or delete packets	Can read packets	Can inject spoofed packets
Man-in-the-middle attacker	✓	✓	✓
On-path attacker		✓	✓
Off-path attacker			✓

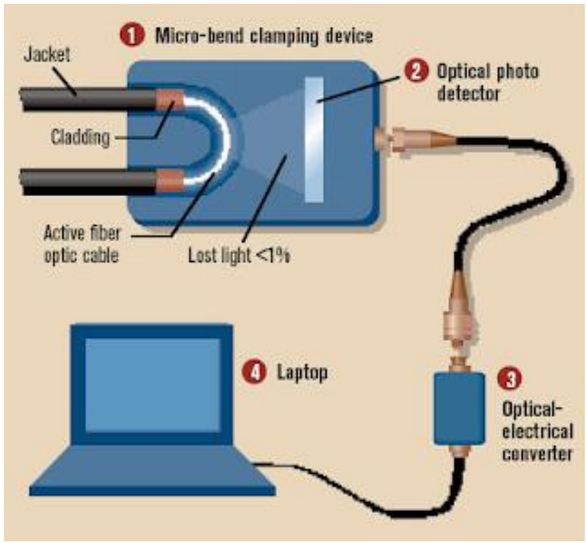
Spoofing



- **Anybody can send their own packets through the network**
- **Spoofing: Lying about the identity of the sender**
  - Example: Mallory sends a message and says the message is from Alice
  - The attacker can lie about the *source address* in the packet header
- **All types of attackers can spoof packets**
  - However, some spoofing attacks may be harder if the attacker can't read or modify packets

Real-World On-Path Attackers

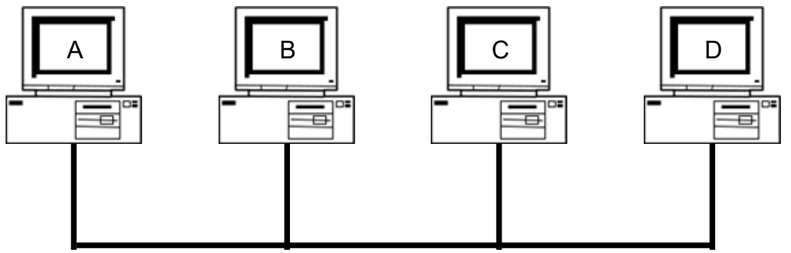
- How might a real-life attacker read packets?
- Layer 1 attack: Use a special device to read bits being transmitted across space





Real-World On-Path Attackers

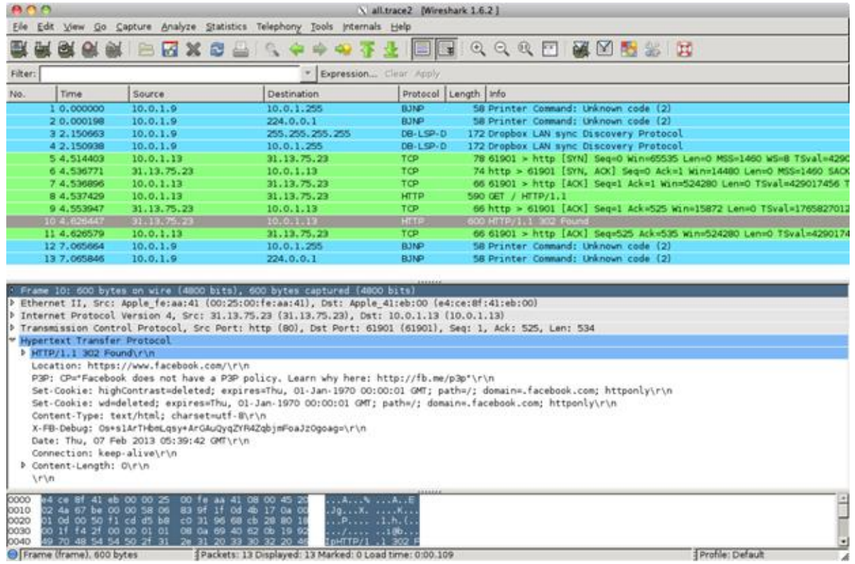
- **Layer 2 attack: Read packets sent across the local area network (LAN)**
- **Recall: A LAN is a network of connected machines**
  - Any machine on the LAN can send packets to any other machine on the LAN
- **Some LANs use broadcast technologies**
  - Every packet gets sent to every machine on the LAN
  - Each machine agrees to ignore packets where the destination is a different machine
- **A machine can break the agreement and read packets meant for other machines**
  - This is called **promiscuous mode**
  - May require root access on the machine



Real-World On-Path Attackers

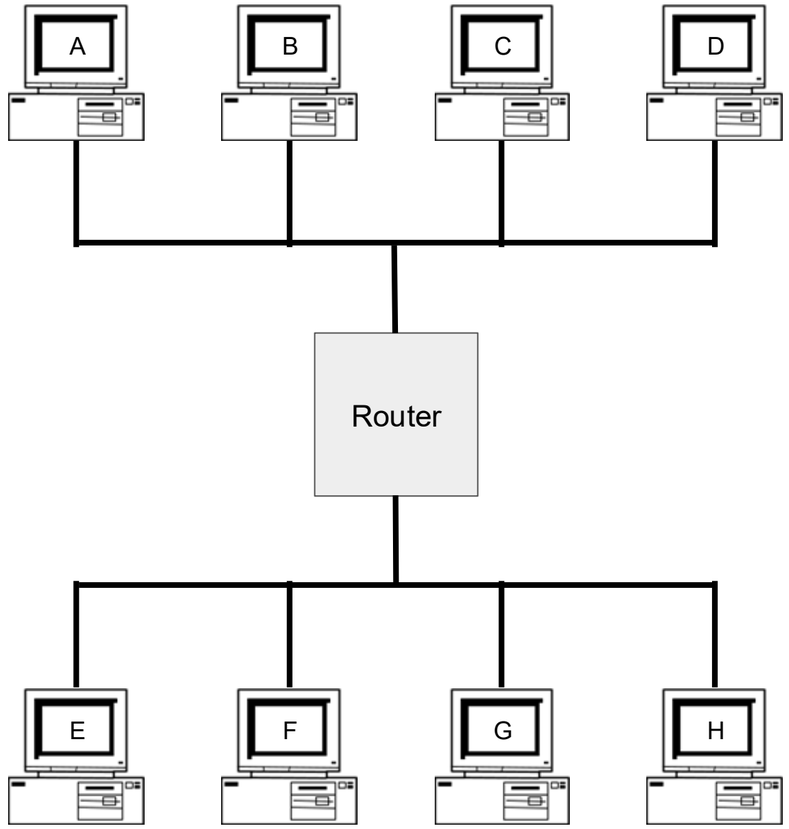
- **tcpdump: A program for reading packets on the local network**
  - Uses promiscuous mode to read other machines' packets in broadcast technologies
- **Wireshark: A graphical user interface (GUI) for analyzing tcpdump packets**

```
demo 2 % tcpdump -r all.trace2
reading from file all.trace2, link-type EN10MB (Ethernet)
21:39:37.772367 IP 10.0.1.9.60627 > 10.0.1.255.canon-bjnp2: UDP, length 16
21:39:37.772565 IP 10.0.1.9.62137 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
21:39:39.923030 IP 10.0.1.9.17500 > broadcasthost.17500: UDP, length 130
21:39:39.923305 IP 10.0.1.9.17500 > 10.0.1.255.17500: UDP, length 130
21:39:42.286770 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [S], seq 2
523449627, win 65535, options [mss 1460,nop,wscale 3,nop,nop,TS val 429017455 ecr 0,sack
OK,eol], length 0
21:39:42.309138 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [S.], seq
3585654832, ack 2523449628, win 14480, options [mss 1460,sackOK,TS val 1765826995 ecr 42
9017455,nop,wscale 9], length 0
21:39:42.309263 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [.], ack 1
, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 0
21:39:42.309796 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [P.], seq
1:525, ack 1, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 524
21:39:42.326314 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [.], ack 5
25, win 31, options [nop,nop,TS val 1765827012 ecr 429017456], length 0
21:39:42.398814 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [P.], seq
1:535, ack 525, win 31, options [nop,nop,TS val 1765827083 ecr 429017456], length 534
21:39:42.398946 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [.], ack 5
35, win 65535, options [nop,nop,TS val 429017457 ecr 1765827083], length 0
21:39:44.838031 IP 10.0.1.9.54277 > 10.0.1.255.canon-bjnp2: UDP, length 16
21:39:44.838213 IP 10.0.1.9.62896 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
```



Review: Layer 2 and Layer 3

- **Local area network (LAN): A set of machines connected in a local network**
  - The MAC address identifies devices on layer 2
- **Internet (IP): Many LANs connected together with routers**
  - The IP address identifies devices on layer 3



Address Resolution Protocol (ARP)



- **ARP: Translates layer 3 IP addresses to layer 2 MAC addresses**
  - Example: Alice wants to send a message to Bob on the local network, but Alice only knows Bob's IP address (**1.2.3.4**). To use layer 2 protocols, she must learn Bob's MAC address.
- **Steps of the protocol**
  1. **Alice checks her cache to see if she already knows Bob's MAC address.**
  2. If Bob's MAC address is not in the cache, Alice **broadcasts** to everyone on the LAN: "What is the MAC address of **1.2.3.4**?"
  3. Bob responds by sending a message only to Alice: "My IP is **1.2.3.4** and my MAC address is **ca:fe:f0:0d:be:ef**." Everyone else does nothing.
  4. **Alice caches Bob's MAC address.**

Address Resolution Protocol (ARP)

arp-normal.pcap — Wireshark-style view, captured on Alice's laptop (filter: arp)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	20:61:84:3a:a9:52	Broadcast	ARP	42	Who has 1.2.3.4? Tell 1.2.3.10
2	0.000812	ca:fe:f0:0d:be:ef	20:61:84:3a:a9:52	ARP	60	1.2.3.4 is at ca:fe:f0:0d:be:ef

```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured
▾ Ethernet II, Src: 20:61:84:3a:a9:52, Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Source: 20:61:84:3a:a9:52
    Type: ARP (0x0806)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 20:61:84:3a:a9:52
    Sender IP address: 1.2.3.10
    Target MAC address: 00:00:00:00:00:00
    Target IP address: 1.2.3.4
```

```
0000  ff ff ff ff ff ff 20 61  84 3a a9 52 08 06 00 01   ······ a·:·R····
0010  08 00 06 04 00 01 20 61  84 3a a9 52 01 02 03 0a   ····· a·:·R····
0020  00 00 00 00 00 00 01 02  03 04                     ··········
```

Ethernet header (14 bytes)

ARP message (28 bytes)

- Request goes to Broadcast:
  every machine on the LAN receives it.
- Target MAC is all zeros:
  that is what Alice is asking for.

Address Resolution Protocol (ARP)



ARP request and reply, field by field (real values)

Layer	Field	Bytes	Request (Alice → everyone)	Reply (Bob → Alice)
Ethernet	Destination MAC	6	**ff:ff:ff:ff:ff:ff (broadcast)**	**20:61:84:3a:a9:52 (Alice only)**
Ethernet	Source MAC	6	**20:61:84:3a:a9:52**	**ca:fe:f0:0d:be:ef**
Ethernet	Type	2	**0x0806 = ARP**	**0x0806 = ARP**
ARP	Hardware type / size	2 + 1	**1 (Ethernet) / 6**	**1 (Ethernet) / 6**
ARP	Protocol type / size	2 + 1	**0x0800 (IPv4) / 4**	**0x0800 (IPv4) / 4**
ARP	Opcode	2	**1 = request**	**2 = reply**
ARP	Sender MAC	6	**20:61:84:3a:a9:52**	**ca:fe:f0:0d:be:ef ← the answer**
ARP	Sender IP	4	**1.2.3.10**	**1.2.3.4**
ARP	Target MAC	6	**00:00:00:00:00:00 (unknown)**	**20:61:84:3a:a9:52**
ARP	Target IP	4	**1.2.3.4 ← the question**	**1.2.3.10**

- Nothing in the reply proves that the sender really owns 1.2.3.4: no signature, no authentication code, no certificate.
- Alice simply stores "1.2.3.4 → ca:fe:f0:0d:be:ef" in her ARP cache. She cannot tell a true reply from a false one.
- The reply is 60 bytes on the wire: 42 bytes of headers + 18 bytes of padding to reach Ethernet's 60-byte minimum.

Address Resolution Protocol (ARP)

Alice knows Bob's IP address (1.2.3.4) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC

Alice

Bob

Charlie

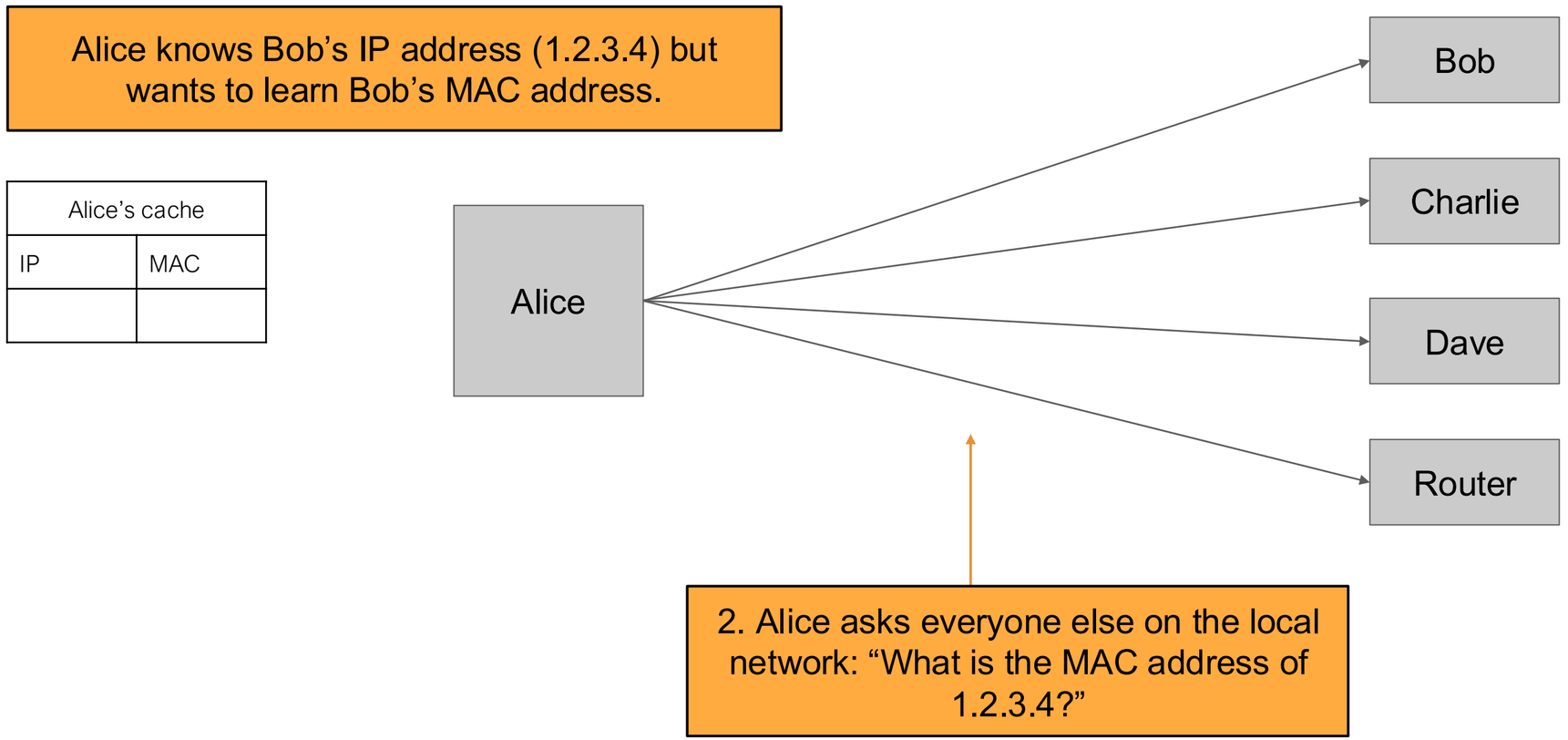
Dave

Router

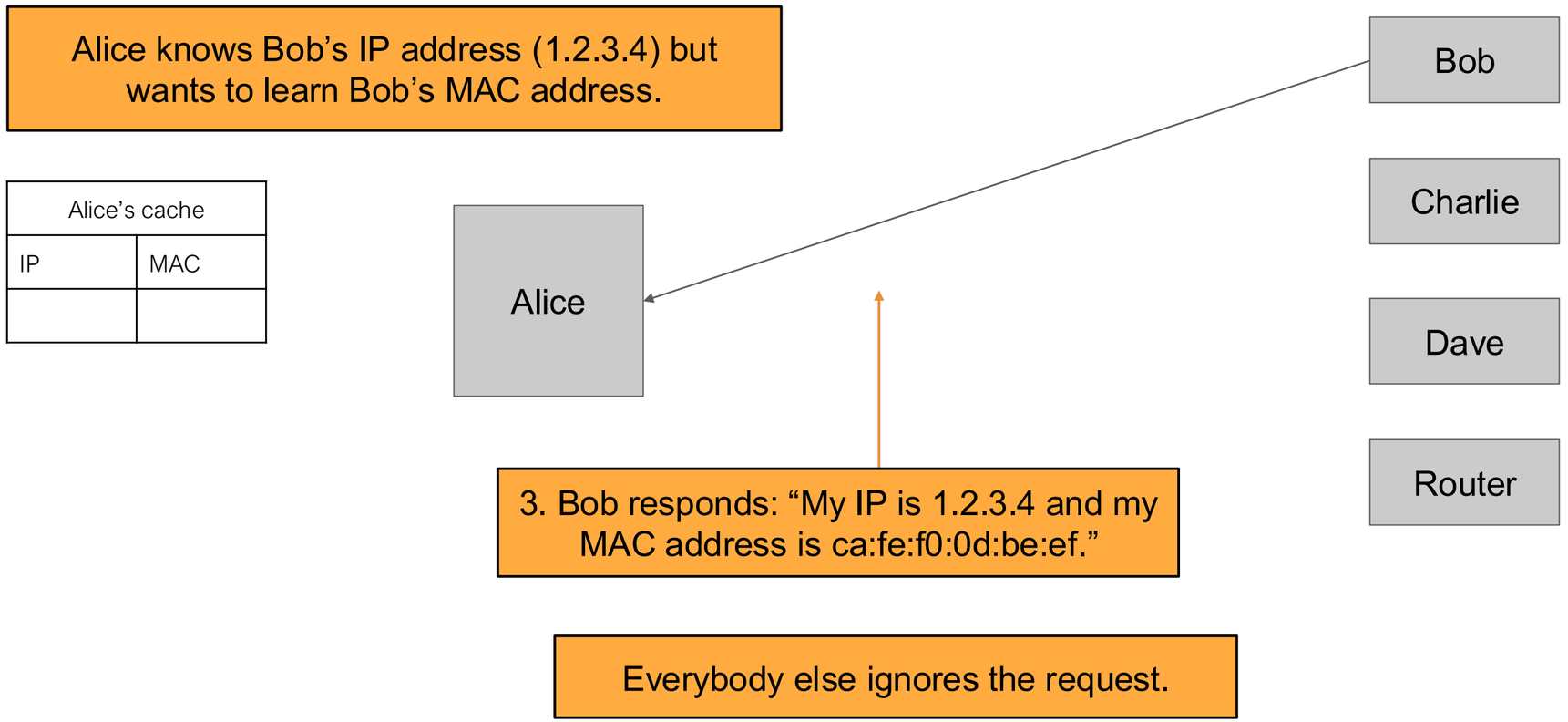
1. Alice checks her cache to see if she already knows the MAC address corresponding to 1.2.3.4.

Since her cache is empty, she must make a request to find out.

Address Resolution Protocol (ARP)



Address Resolution Protocol (ARP)



Address Resolution Protocol (ARP)



Alice knows Bob's IP address (1.2.3.4) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC
`1.2.3.4`	`ca:fe:f0:0d:be:ef`

Alice

Bob

Charlie

Dave

Router

4. Alice adds Bob's MAC address to her cache.

Address Resolution Protocol (ARP)



- **If Bob is outside of the LAN, the router will respond with its own MAC address**
  - If Alice wants to send a packet to Bob, she sends the packet to the router
  - The router can forward the packet to other LANs to reach Bob
  - Alternatively, if Alice knows what addresses belong to the LAN, she will request the MAC address that corresponds to the router's IP
- **All received ARP replies are cached, even if no request was sent**

Attacks on ARP

Alice knows Bob's IP address (1.2.3.4) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC

Alice

Bob

Charlie

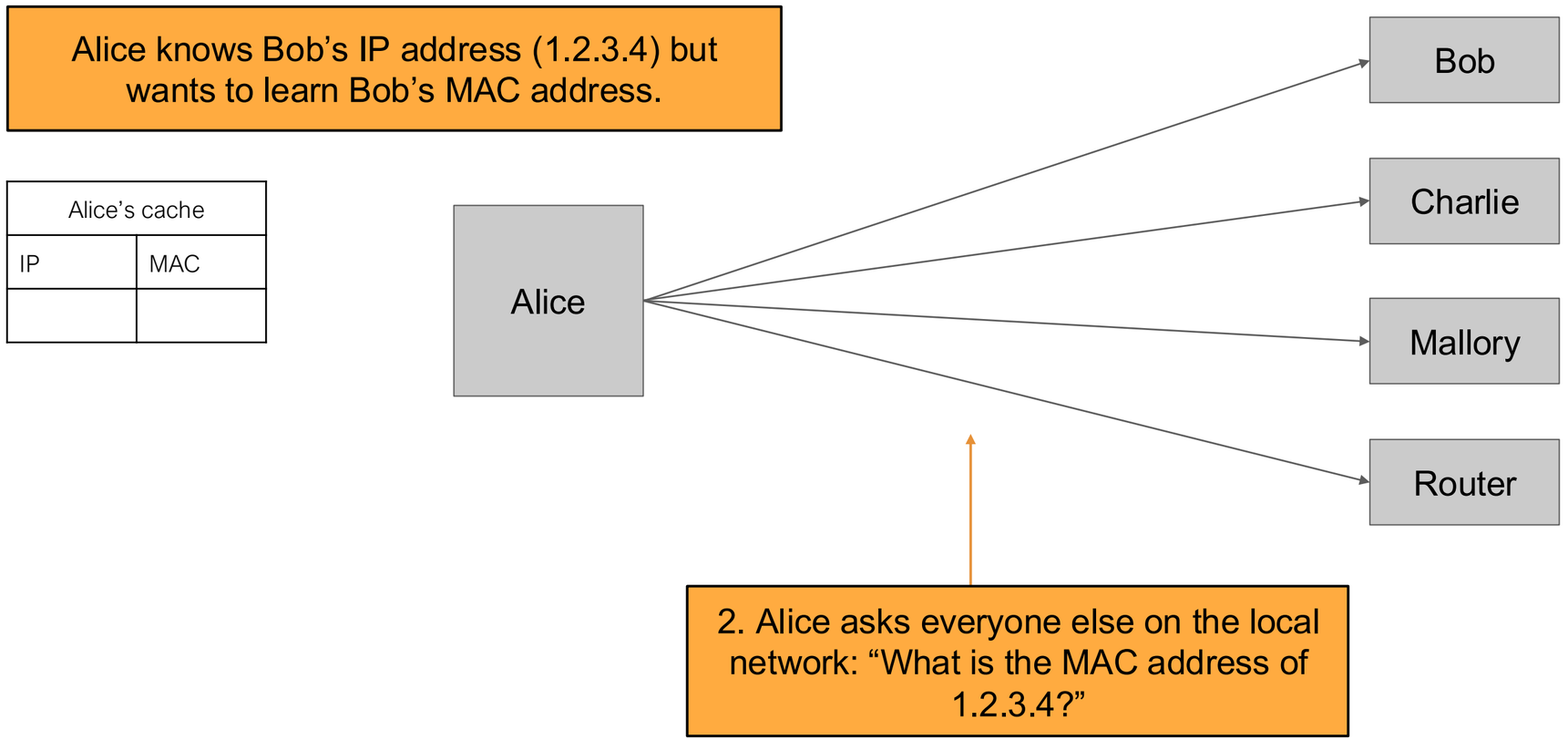
Mallory

Router

1. Alice checks her cache to see if she already knows the MAC address corresponding to 1.2.3.4.

Since her cache is empty, she must make a request to find out.

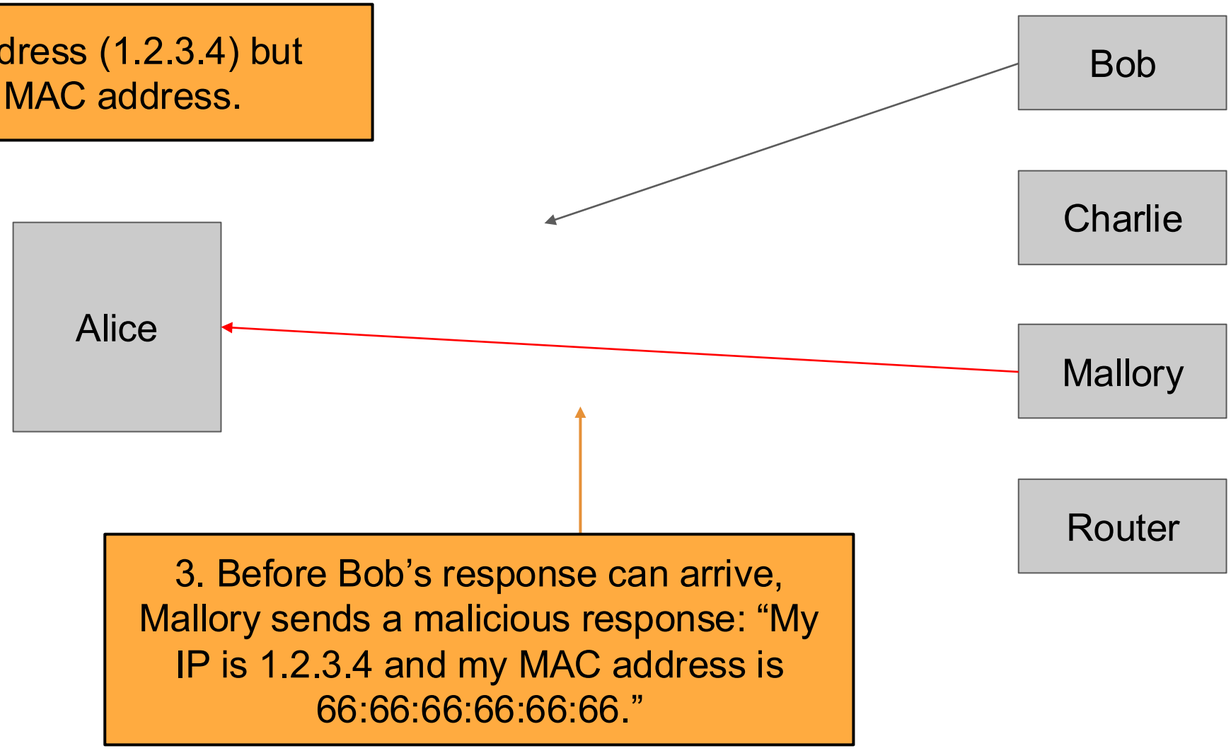
Attacks on ARP



Attacks on ARP

Alice knows Bob's IP address (1.2.3.4) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC



3. Before Bob's response can arrive, Mallory sends a malicious response: "My IP is 1.2.3.4 and my MAC address is 66:66:66:66:66:66."

Attacks on ARP

Alice knows Bob's IP address (1.2.3.4) but wants to learn Bob's MAC address.

Alice's cache	
IP	MAC
1.2.3.4	66:66:66:66:66:66

Alice

Bob

Charlie

Mallory

Router

4. Alice adds Mallory's malicious address to her cache.

Attacks on ARP

arp-spoofing.pcap — captured on Alice's laptop

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	20:61:84:3a:a9:52	Broadcast	ARP	42	Who has 1.2.3.4? Tell 1.2.3.10
2	0.000231	66:66:66:66:66:66	20:61:84:3a:a9:52	ARP	60	1.2.3.4 is at 66:66:66:66:66:66
3	0.000812	ca:fe:f0:0d:be:ef	20:61:84:3a:a9:52	ARP	60	1.2.3.4 is at ca:fe:f0:0d:be:ef (duplicate use of 1.2.3.4 detected!)
4	0.000955	1.2.3.10	1.2.3.4	TCP	66	51514 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460

Wireshark notices two different MACs for one IP and marks frame 3 (yellow). Alice's operating system does not: it has already cached the first answer.

Frame 2 and frame 3 side by side

	Frame 2: Mallory (forged)	Frame 3: Bob (genuine)
Arrives after	0.231 ms	0.812 ms
Ethernet source	**66:66:66:66:66:66**	**ca:fe:f0:0d:be:ef**
ARP opcode	2 (reply)	2 (reply)
Sender IP	**1.2.3.4**	**1.2.3.4**
Sender MAC	**66:66:66:66:66:66**	**ca:fe:f0:0d:be:ef**
Alice's ARP cache	1.2.3.4 → 66:66:66:66:66:66 (stored)	ignored: < 1 s after the first reply

Frame 4: Alice now connects to Bob — and the frame goes to Mallory

Layer	Field	Value	Meaning
Ethernet	Destination MAC	**66:66:66:66:66:66**	Mallory's MAC, taken from the forged reply
Ethernet	Source MAC	**20:61:84:3a:a9:52**	Alice
IPv4	Destination IP	**1.2.3.4**	still Bob's IP address
TCP	Destination port	**80**	Alice is opening an HTTP connection to Bob

- The IP header says "to Bob", the Ethernet header says "to Mallory". On a LAN only the Ethernet header decides who receives the frame.
- Mallory reads it and forwards it to Bob (ca:fe:f0:0d:be:ef): Alice and Bob notice nothing.
- Linux ignores a second reply that arrives within 1 s of the first ("locktime"), so in a race the first answer wins.
- Later replies do overwrite the entry, so Mallory keeps re-sending forged replies every second or two.

Attack: ARP Spoofing



- **Alice has no way of verifying the ARP response**
  - Spoofing: Any attacker on the local network can claim to have the requested IP address
- Alice is only expecting one machine to respond, so she will accept the first response
  - **Race condition**: As long as the attacker responds faster, the requester will accept the attacker's response
- **ARP spoofing requires Mallory to be in the same LAN as Alice**
- **ARP spoofing lets Mallory become a man-in-the-middle (MITM) attacker**
  - Alice thinks that Bob's MAC address is **66:66:66:66:66:66** (Mallory's MAC address)
  - When Alice sends a message to Bob, she is actually sending the message to Mallory
  - Mallory can modify the message and then send the modified message to Bob

ARP Spoofing: Defenses



- **Use switches (with ARP Suppression) to avoid broadcasts and ARP requests**
  - When Alice wants to send a message to Bob, she sends the message to a switch on the LAN
  - The switch maintains a cache of IP/MAC mappings
    - Since everyone is using the switch, its cache is larger than an individual user's cache
  - If Bob's MAC address is in the cache, the switch sends the message directly to Bob
  - Otherwise, the switch broadcasts the message
- **Benefits of switches**
  - Security: Reduces the number of messages broadcast to the entire LAN
  - Efficiency: Fewer broadcast requests means more requests can be sent per second
  - Isolation: Smarter switches implement virtual local area networks (VLANs), which split a LAN into several isolated parts
    - One part of the VLAN cannot directly interact with other parts of the VLAN
- Tools like **arpwatch** track ARP responses and make sure that there is no suspicious activity

ARP Spoofing: More Defenses



- **Static ARP entries**
  - The administrator fixes the IP→MAC mapping for critical hosts (e.g., the gateway) on each user device
  - The host ignores ARP replies for that IP, so forged replies cannot change it
  - Drawback: does not scale; must be updated manually when hardware changes
- **Dynamic ARP Inspection (DAI) on the switch**
  - The switch learns legitimate IP–MAC pairs from DHCP (the "DHCP snooping" table)
  - It drops any ARP reply whose IP–MAC pair is not in that table
  - Stops the attacker's forged replies before they reach the victim

ARP Spoofing: More Defenses



- **What does NOT stop ARP spoofing**
  - Encryption (e.g., TLS): protects the content, but frames still go to the attacker
  - Changing your IP address: the attacker simply forges replies for the new address
    - For instance, let's say you change your IP from 192.168.1.10 to 192.168.1.50
    - When someone wants to connect you, it will send an ARP saying who is 192.168.1.50?
    - Then the attacker can still use ARP spoofing to respond that he/she is the 192.168.1.50



Dynamic Host Configuration Protocol (DHCP)

DHCP: Initial Network Configuration



- **To connect to a network, a user needs:**
  - An IP address so that other people can contact the user
  - The IP address of the DNS server (we'll see this soon)
  - The IP address of the router (gateway) so that the user can contact machines outside of the LAN
- **The first time a user connects, they don't have this information yet**
  - The user also doesn't know who to ask for this information
- **DHCP gives the user a configuration when they first join the network**

Steps of the DHCP Handshake



1. **Client Discover: The client broadcasts a request for a configuration**
2. **DHCP Offer: Any DHCP server can respond with a configuration offer**
   - Usually only one DHCP server responds
   - The offer includes an IP address for the client, the DNS server's IP address, and the (gateway) router's IP address
   - The offer also has an expiration time (how long the user can use this configuration)
3. **Client Request: The client broadcasts which configuration it has chosen**
   - If multiple DHCP servers made offers, the ones that were not chosen discard their offer
   - The chosen DHCP server gives the offer to the client
4. **DHCP Acknowledgement**: The chosen server confirms that its configuration has been given to the client

Dynamic Host Configuration Protocol (DHCP)



Alice's configuration	
My IP	???
DNS Server	???
Gateway	???

Alice

Bob

DHCP Server 1

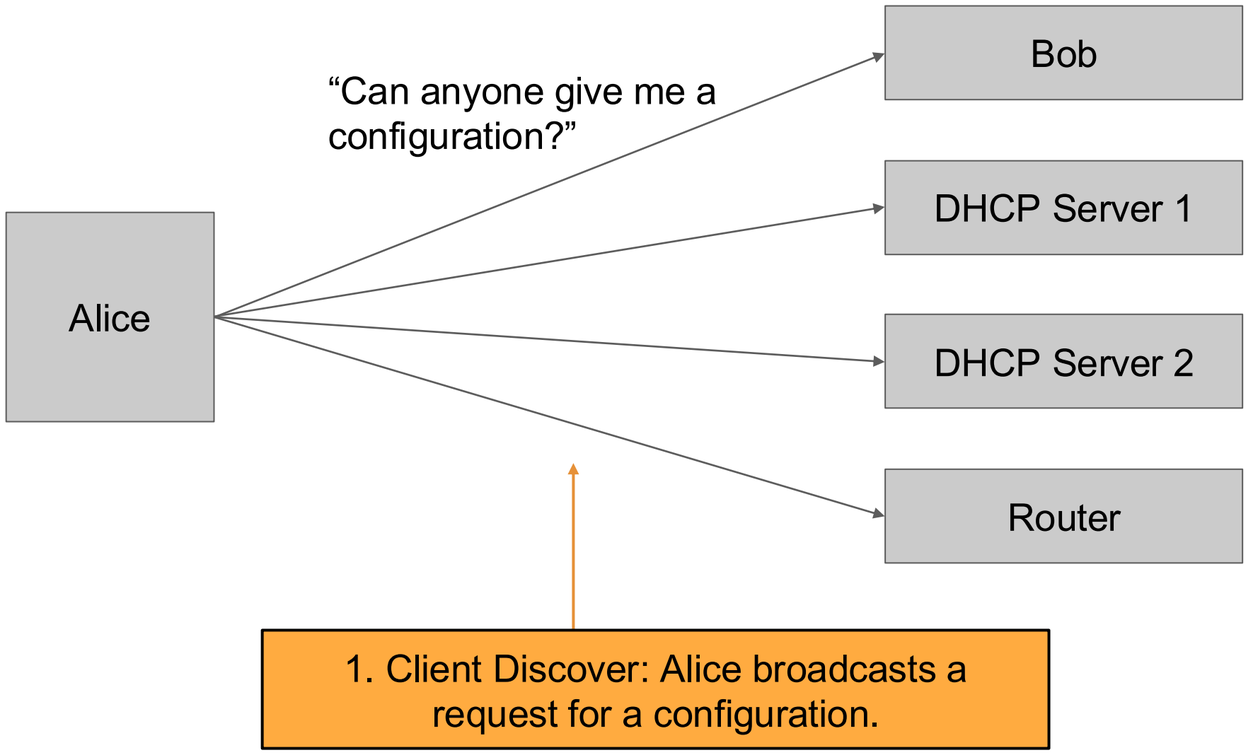
DHCP Server 2

Router

Alice wants to connect to the network, but she's missing a configuration.

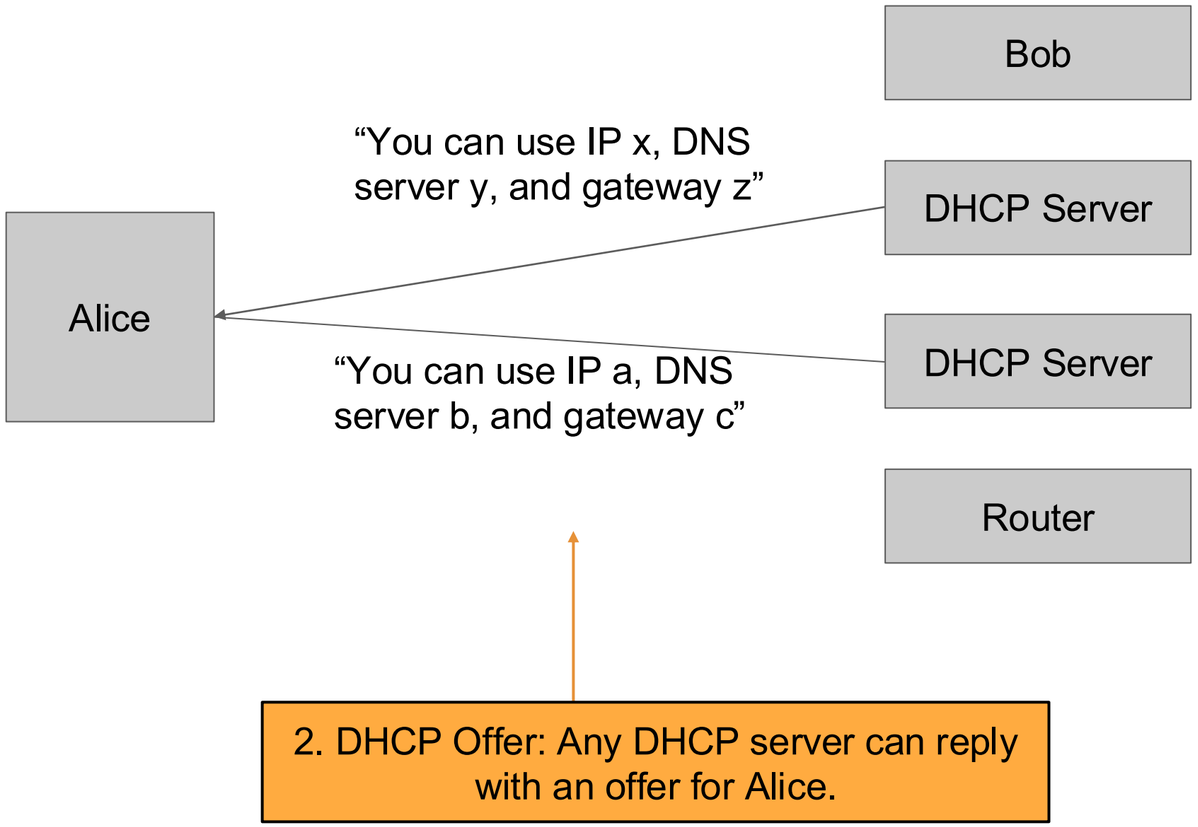
Dynamic Host Configuration Protocol (DHCP)

Alice's configuration	
My IP	???
DNS Server	???
Gateway	???



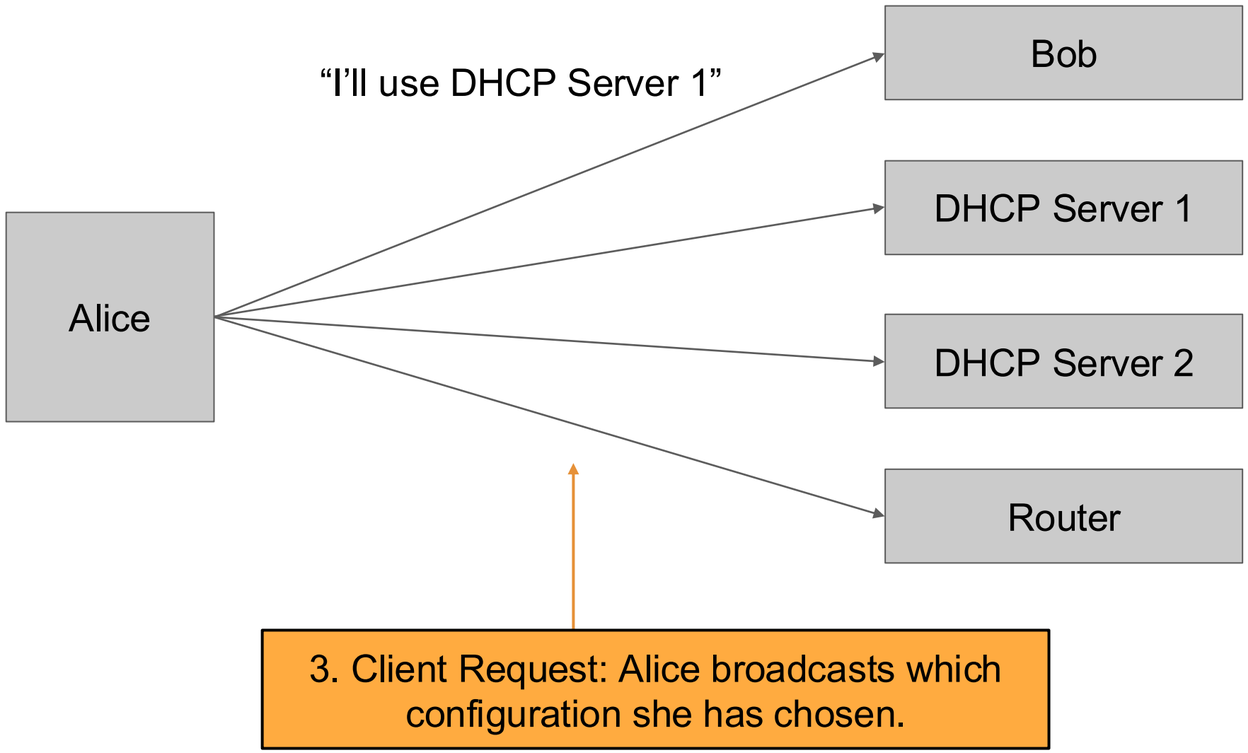
Dynamic Host Configuration Protocol (DHCP)

Alice's configuration	
My IP	???
DNS Server	???
Gateway	???

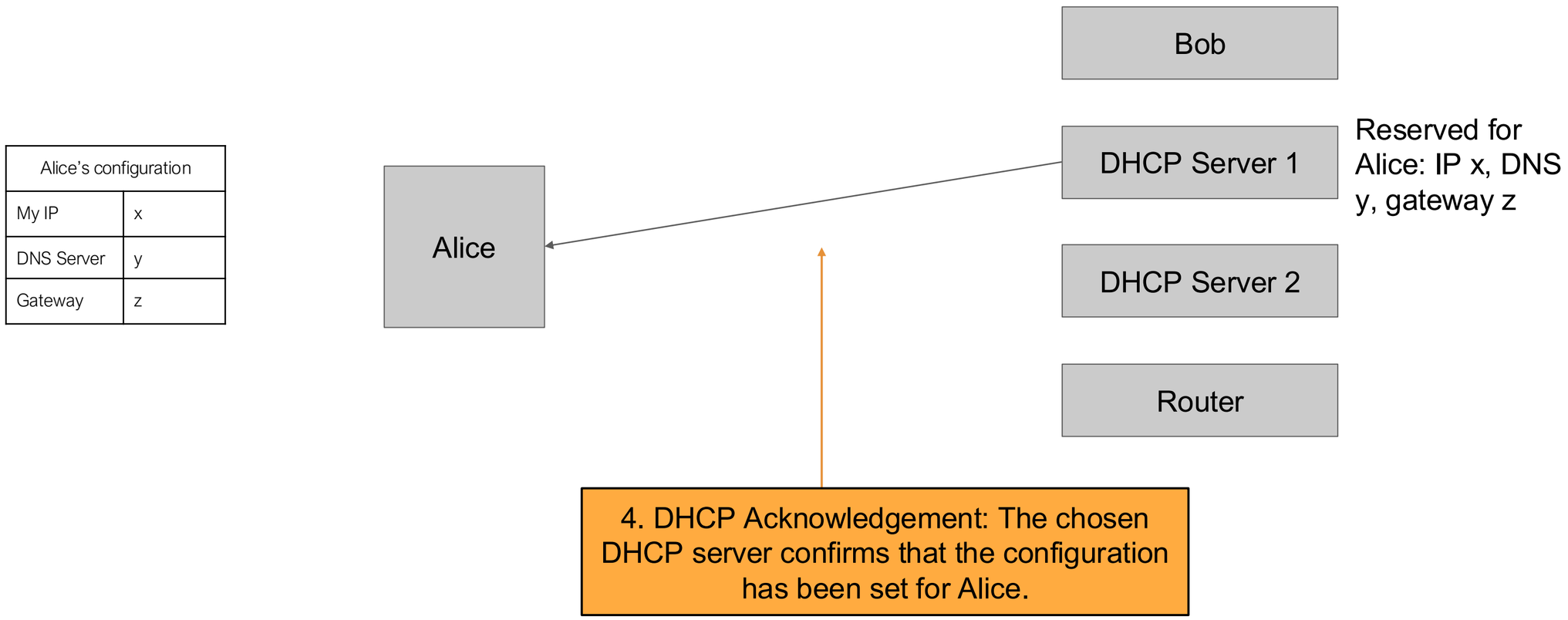


Dynamic Host Configuration Protocol (DHCP)

Alice's configuration	
My IP	???
DNS Server	???
Gateway	???



Dynamic Host Configuration Protocol (DHCP)



Dynamic Host Configuration Protocol (DHCP)

dhcp-normal.pcap — Alice's laptop joins the network (filter: dhcp)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	316	DHCP Discover - Transaction ID 0x3903f326
2	0.002104	1.2.3.2	255.255.255.255	DHCP	316	DHCP Offer - Transaction ID 0x3903f326
3	0.002391	0.0.0.0	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0x3903f326
4	0.004017	1.2.3.2	255.255.255.255	DHCP	316	DHCP ACK - Transaction ID 0x3903f326

```
▸ Frame 2: 316 bytes on wire
▾ Ethernet II, Src: 00:1b:21:3a:4f:02, Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Internet Protocol Version 4, Src: 1.2.3.2, Dst: 255.255.255.255
▾ User Datagram Protocol, Src Port: 67, Dst Port: 68
▾ Dynamic Host Configuration Protocol (Offer)
  Message type: Boot Reply (2)
  Transaction ID: 0x3903f326
  Bootp flags: 0x8000, Broadcast flag (Broadcast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 1.2.3.10
  Client MAC address: 20:61:84:3a:a9:52
  ▸ Option: (53) DHCP Message Type (Offer)
  ▸ Option: (54) DHCP Server Identifier (1.2.3.2)
  ▸ Option: (51) IP Address Lease Time (86400s) 1 day
  ▸ Option: (1) Subnet Mask (255.255.255.0)
  ▾ Option: (3) Router
      Router: 1.2.3.1
  ▾ Option: (6) Domain Name Server
      Domain Name Server: 1.2.3.53
  ▸ Option: (255) End
```

- Discover and Request come from 0.0.0.0 (Alice has no address yet) and go to 255.255.255.255 (everyone).
- The highlighted lines are the whole configuration: Alice's IP, the subnet mask, the lease time, the router and the DNS server.
- The offer is tied to Alice's Discover only by the Transaction ID and her MAC address — and every machine on the LAN saw both.

DHCP Attacks



- **Alice has no way of verifying the DHCP response**
  - Spoofing: Any attacker on the local network can claim to have a configuration
- Alice usually expects only one DHCP server to respond, so she will accept the first response
  - **Race condition**: As long as the attacker responds faster, Alice will accept the attacker's response
- **DHCP attacks require Mallory to be in the same LAN as Alice**
- **DHCP attacks let Mallory become a man-in-the-middle (MITM) attacker**
  - Mallory claims the gateway router's address is Mallory's address
    - When Alice tries to send a message to the rest of the Internet, she actually sends it to Mallory
    - Mallory can modify the message before sending it to its destination
  - Mallory can also claim the DNS server's address is Mallory's address

DHCP Attack

dhcp-spoofing.pcap — two servers answer; Alice takes the first offer

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	316	DHCP Discover - Transaction ID 0x3903f326
2	0.000690	1.2.3.66	255.255.255.255	DHCP	316	DHCP Offer - Transaction ID 0x3903f326
3	0.002104	1.2.3.2	255.255.255.255	DHCP	316	DHCP Offer - Transaction ID 0x3903f326
4	0.002391	0.0.0.0	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0x3903f326
5	0.002988	1.2.3.66	255.255.255.255	DHCP	316	DHCP ACK - Transaction ID 0x3903f326

Red = sent by Mallory. Frame 4: Alice's Request names server 1.2.3.66 (Option 54), so the real server withdraws its offer.

The two offers side by side

	Frame 2: Mallory's offer (accepted)	Frame 3: real server's offer
Arrives after	0.7 ms	2.1 ms
Sent from (Ethernet / IP)	66:66:66:66:66:66 / 1.2.3.66	00:1b:21:3a:4f:02 / 1.2.3.2
Transaction ID	**0x3903f326**	**0x3903f326**
Your IP address	1.2.3.10	1.2.3.10
Subnet mask, lease	255.255.255.0, 1 day	255.255.255.0, 1 day
Router (gateway)	**1.2.3.66 = Mallory**	**1.2.3.1**
DNS server	**1.2.3.66 = Mallory**	**1.2.3.53**

- Mallory copied the Transaction ID from Alice's broadcast Discover, so her offer matches.
- Her offer looks normal: a valid address in the right subnet. Only two configuration values differ — the router and the DNS server.
- Result: every packet Alice sends off the LAN goes to Mallory first, and Mallory answers all of Alice's DNS lookups.
- Mallory is now a man-in-the-middle for Alice's Internet traffic, for the whole lease (1 day).

ARP and DHCP



- **The attacks on ARP and DHCP are very similar**
  - Spoofing: The attacker claims to have an answer
  - Race condition: The requester accepts the first response. As long as the attacker's response arrives first, it is accepted
- **Main vulnerabilities**
  - Broadcast protocols: Requests are sent to everyone on the LAN, so the attacker can see every request
  - No trust anchor: There is no way to verify that responses are legitimate

DHCP Defenses



- **DHCP attacks are hard to defend against**
  - No root of trust: When we first connect, there's nobody we can trust
- **Instead, we rely on defenses provided in higher layers**

DHCP Defenses



- Let’s say the malicious DHCP server assigns a fake gateway/DNS to you
- What you access https://bank.com
- The fake DNS returns that bank.com -> 10.x.x.x (controlled by hacker)
- Then you will access 10.x.x.x
- However, the certificate on the hacker’s server is invalid – since the hacker cannot get the private key to sign the certificate for bank.com

One example of relying high level protocol to defend the attack



Wireless Local Networks

Wi-Fi



- **Wi-Fi: A layer 2 protocol that wirelessly connects machines in a LAN**
  - Alternative is Ethernet, which uses wires to connect machines in a LAN
- **Parts of a Wi-Fi network**
  - **Access point**: A machine that will help you connect to the network
  - **SSID**: The name of the Wi-Fi network
  - Password: Optionally, a password to secure Wi-Fi communications

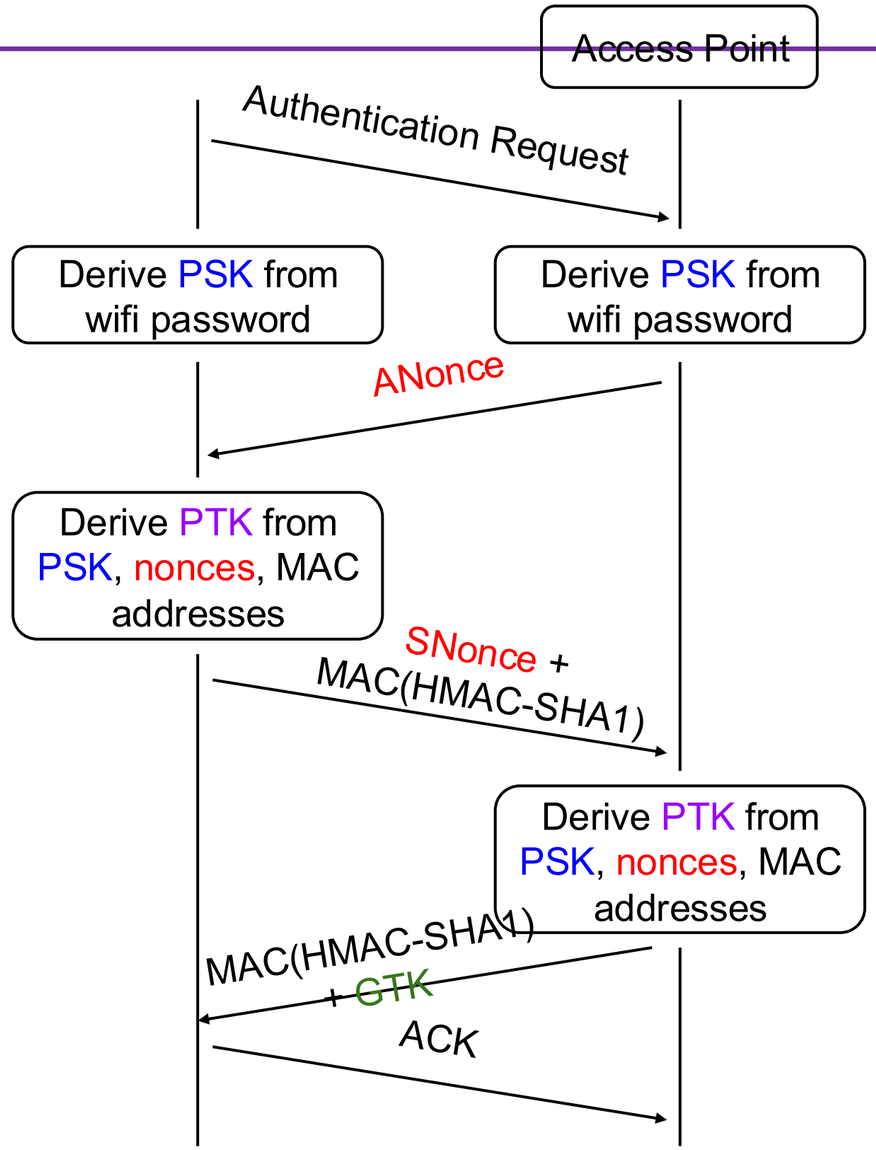
WPA2



- **Wi-Fi Protected Access 2** (**WPA2**): A protocol for securing Wi-Fi network communications with cryptography
- **Design goals**
  - **Everyone with the Wi-Fi password can join the network**
  - Messages sent over the network are encrypted with keys
  - An attacker who does not know the Wi-Fi password cannot learn the keys

Client: Optimized WPA 4-Way Handshake

1. **The client sends an authentication request to the access point**
2. **Both use the password to derive the *PSK* (pre-shared key)**
3. **The AP sends *ANonce* to the client**
4. The client generates *SNonce*, uses the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
5. **The client sends *SNonce* and its MAC to the AP**
6. The AP uses the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
7. **The AP sends its MAC and *GTK* to the client**
8. **The client acknowledges receiving the GTK**



WPA Handshake



- **Both sides derive secret keys for communication**
  - Wi-Fi password → *PSK*
  - *PSK* + nonces + MAC addresses → *PTK*
  - The *PTK* is used to encrypt and authenticate all future communication
  - Note: The PTK is different **for every user, because of the nonces**
- **The access point encrypts and sends the *GTK* to the client**
  - The GTK is used for messages broadcast to the entire network
  - Everyone on the network uses the same GTK

PTK



- MIC (Message Integrity Code): HMAC-SHA1
- Remember HMAC-SHA1?
  - **HMAC(*K*, *M*) = *H*((*K*' ⊕ *opad*) || H((*K*' ⊕ *ipad*) || *M*))**
  - *opad* (outer pad) is the hard-coded byte **0x5c** repeated until it's the same length as *K*'
  - *ipad* (inner pad) is the hard-coded byte **0x36** repeated until it's the same length as *K*'

The PTK is cut into three keys:

Key	Name	Part of PTK	Value	Used for
KCK	Key Confirmation Key	bytes 0–15	98 06 03 39 b2 e4 24 ae …	computes the MIC of messages 2, 3, 4
KEK	Key Encryption Key	bytes 16–31	fa a7 89 cf 57 8f 4d 84 …	encrypts the GTK in message 3
TK	Temporal Key	bytes 32–47	a7 44 1e 65 36 5d 87 a4 …	encrypts all data frames afterwards (AES-CCMP)

Review: HMAC-SHA1



- Remember HMAC-SHA1?
  - **HMAC(*K*, *M*) = *H*((*K*' ⊕ *opad*) || H((*K*' ⊕ *ipad*) || *M*))**
  - HMAC provides integrity, authenticity?
- Quick question: why we cannot use SHA-1(Key || m) for MAC? – SHA1 is vulnerable to length extension attack – because its construction (SHA-1, MD5, and SHA-2 all vulnerable)
  - Let's say the attacker get message m= "user=alice" and H = SHA-1(Key || m)
  - The attacker does not know the kye. However, he/she can compute SHA-1(Key || "user=alice" || padding || "&role=admin") by using **H as the initial state of the SHA-1 function**
  - If an attacker knows the hash H(M) of a secret message M, they can compute the hash of M || padding || extra data without knowing M

GTK



- The GTK is used for messages broadcast to the entire network: Why
  - That’s because we need to broadcast messages to the entire network
  - ARP message!

From Wi-Fi password to keys

From the Wi-Fi password to the session keys (values computed for this capture)

```
password   = "lowlevel4130"
SSID       = "IERG4130-Lab"
```

PBKDF2 →

PMK (32 bytes) — also called the PSK

`60 42 7c 2b d6 af 15 01 3a 2e bd a1 fc fa 8c db …`

PBKDF2-HMAC-SHA1 with 4096 rounds: deliberately slow, to make password guessing expensive. Same password + SSID → same PMK.

```
PMK
AP MAC     = 02:1a:11:f0:0d:66
client MAC = 20:61:84:3a:a9:52
ANonce     = 7d3a91c0e2f45b66…
SNonce     = e1b47c2a90d35f68…
```

PRF-384 →

PTK (48 bytes) — Pairwise Transient Key

KCK	`98 06 03 39 b2 e4 24 ae e5 b1 7c 40 d3 6a cc 5c`
KEK	`fa a7 89 cf 57 8f 4d 84 35 2c 08 e6 a3 6a 50 02`
TK	`a7 44 1e 65 36 5d 87 a4 2d 54 70 81 7f fd b6 14`

PRF-384 = HMAC-SHA1 applied 3 times with the label "Pairwise key expansion". New nonces → a new PTK every session.

The PTK is cut into three keys:

Key	Name	Part of PTK	Value	Used for
KCK	Key Confirmation Key	bytes 0–15	`98 06 03 39 b2 e4 24 ae …`	computes the MIC of messages 2, 3, 4
KEK	Key Encryption Key	bytes 16–31	`fa a7 89 cf 57 8f 4d 84 …`	encrypts the GTK in message 3
TK	Temporal Key	bytes 32–47	`a7 44 1e 65 36 5d 87 a4 …`	encrypts all data frames afterwards (AES-CCMP)

- Both sides run exactly these computations on their own. They end up with the same PTK because they know the same password and have seen the same MAC addresses and nonces.
- Mallory's evil twin knows the password too, so she computes this same PTK, and she can decrypt everything Alice sends.

Message 2

Inside message 2: the EAPOL-Key frame (121 bytes, real values)

```
0000  01 03 00 75 02 01 0a 00  00 00 00 00 00 00 00 00  ···u·····  ·····
0010  01 e1 b4 7c 2a 90 d3 5f  68 b2 c4 1e 7d 0a 9f 3b  ···|*··_h···}··;
0020  56 4d 8e 21 c7 f0 a6 b3  9e 5c 1d 74 a2 8f 60 b3  VM·!·····\·t··`·
0030  e9 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ·····  ·····
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ·····  ·····
0050  00 e6 79 c2 cc a4 5f c2  ca 04 b4 be 0b 48 a5 a2  ··y···_·····H··
0060  96 00 16 30 14 01 00 00  0f ac 04 01 00 00 0f ac  ···0·····  ·····
0070  04 01 00 00 0f ac 02 00  00                       ·····  ·····
```

EAPOL header (4 B, offset 0)
version 1, type 3 (Key), length 117

Key Information (2 B, offset 5)
0x010a: pairwise, MIC present, HMAC-SHA1

Replay Counter (8 B, offset 9)
1 (same as message 1)

IV, RSC, Key ID (32 B, offset 49)
all zero in message 2

Key Data Length (2 B, offset 97)
22

Descriptor type (1 B, offset 4)
2 = RSN (WPA2)

Key Length (2 B, offset 7)
0

Key Nonce (32 B, offset 17)
SNonce (32 random bytes)

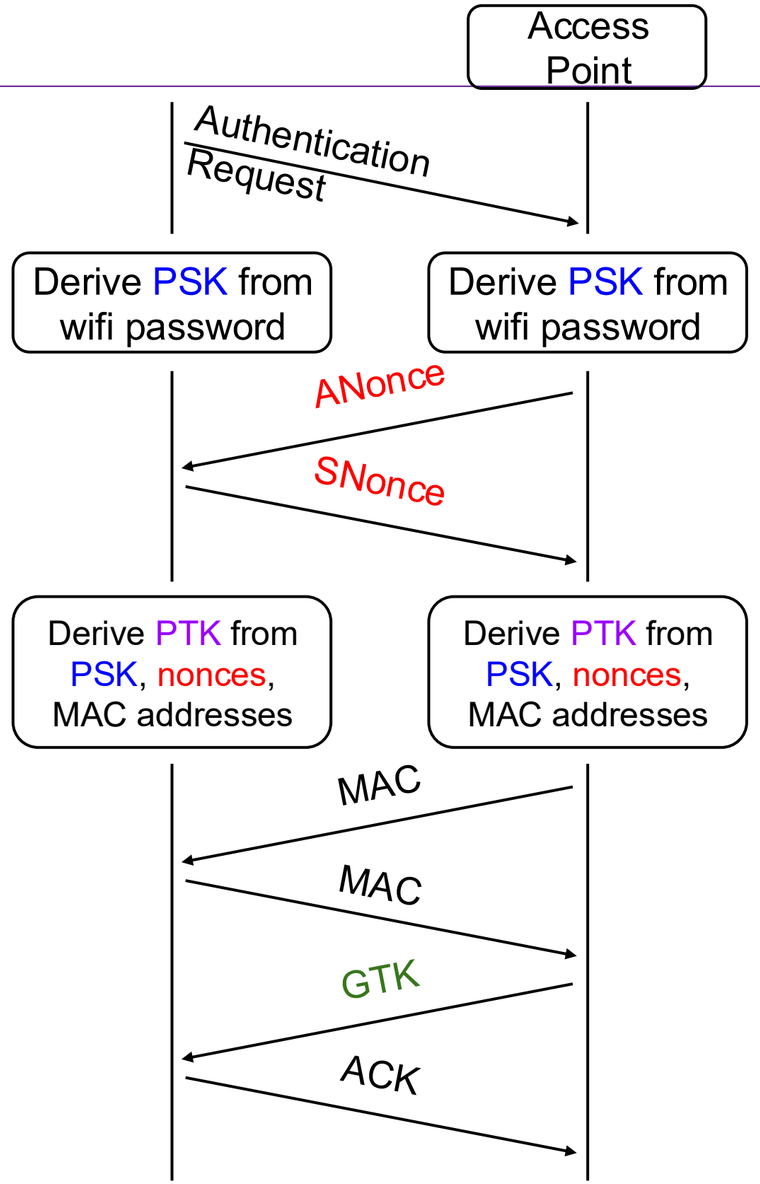
Key MIC (16 B, offset 81)
16 bytes: the MIC

Key Data (22 B, offset 99)
client's RSN element: WPA2, CCMP, PSK

- The two fields that matter: the SNonce (green), which the AP needs to compute the keys, and the MIC (red), which proves the frame came from someone who knows the password.
- Everything here is sent in the clear: anyone nearby can capture it. The password and the keys are never sent.

Client: WPA-PSK Attacks

- **Rogue AP: Pretend to be an AP, and offer your own *ANonce* to the client**
  - If you know the password/PSK, you can complete the 4-way handshake with the client and become a MITM!



Rogue Access Points (Evil Twin)



- **Many networks are protected by one shared password that every user knows**
  - (home Wi-Fi, dorms, cafés)
- An attacker who knows the password can set up an access point with the same network name (SSID) and the same password
- Devices connect to the strongest signal, so victims may join the attacker's AP
- Key point: a shared password proves “you know the password” NOT "this is the real access point"
- Everyone who knows the password, including the attacker, looks the same
- Result: the attacker becomes a man-in-the-middle for all of the victim's traffic

Evil Twin: What Can the Attacker See?



- **Plain HTTP: everything**
  - URLs, form data, passwords, cookies; the attacker can also modify pages
- HTTPS (with proper certificate checking): NOT the content
  - The attacker cannot present a valid certificate for the real site
  - Page content, passwords and cookies stay confidential and integrity-protected

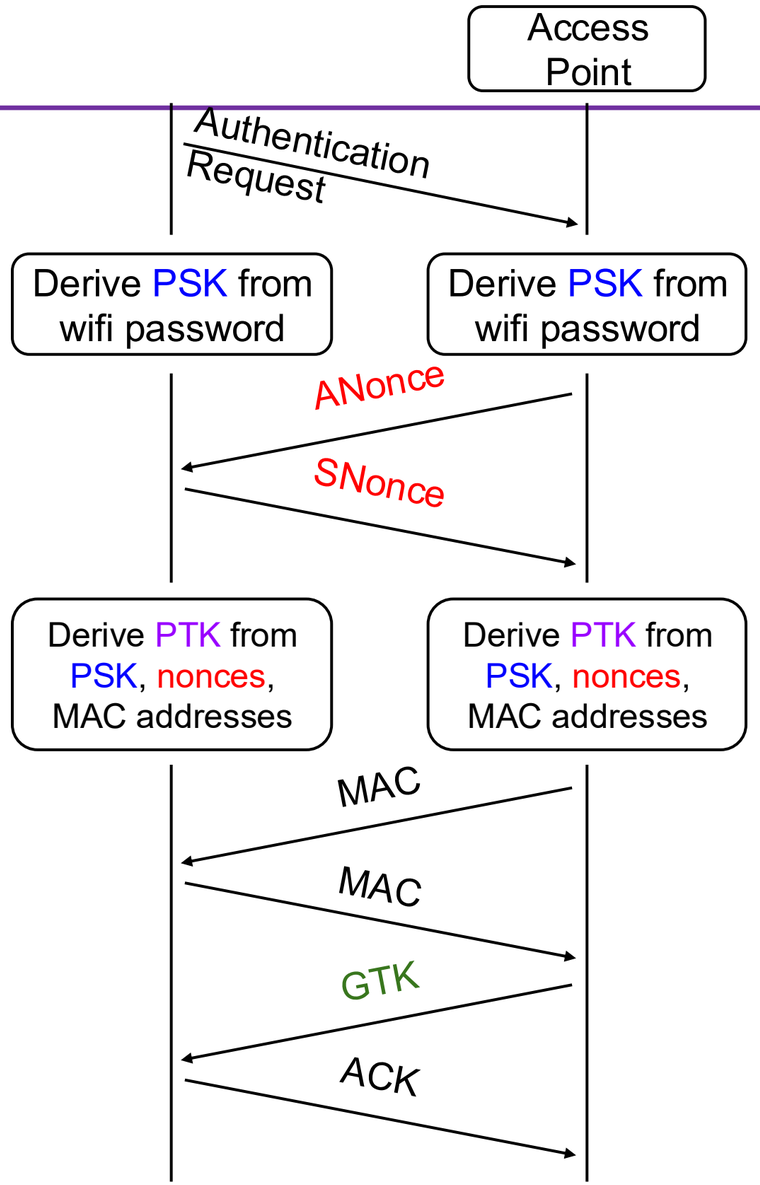
Evil Twin: What Can the Attacker See?



- HTTPS still leaks metadata
  - IP headers are not encrypted, so the attacker sees which server IP you talk to
  - Also: timing and traffic volume (and, typically, DNS queries and the server name)
- Lesson: link-layer protection with a shared password does not give end-to-end security;
  - use end-to-end encryption (TLS) and verify certificates

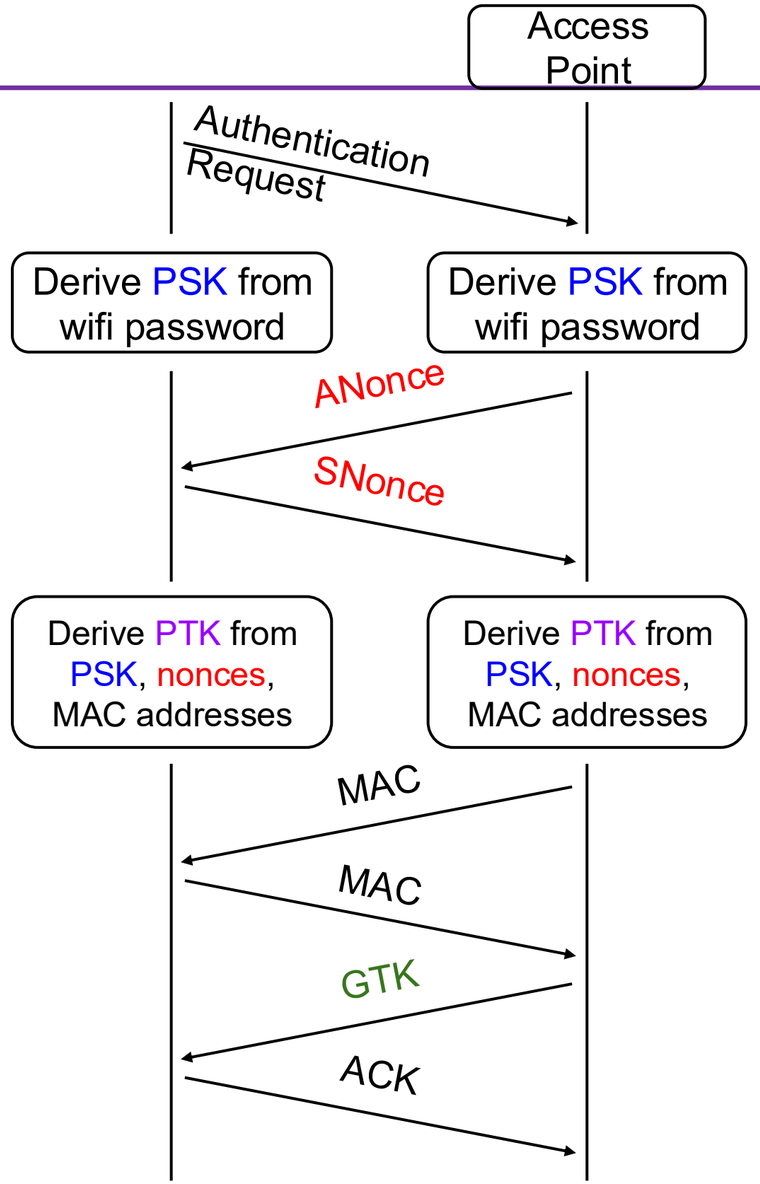
Client: WPA-PSK Attacks

- **Offline brute-force attack**: People tend to choose bad passwords, and you have enough information to know if you guessed the password correctly
  - Nonces are sent unencrypted, and client and AP MAC addresses are public
  - Eavesdropper guesses a password and derives:
    - Wi-Fi password → *PSK*
    - *PSK* + nonces + MAC addresses → *PTK*
    - Eavesdropper checks that the MAC from the guess matches the MAC that was sent



Client: WPA-PSK Attacks

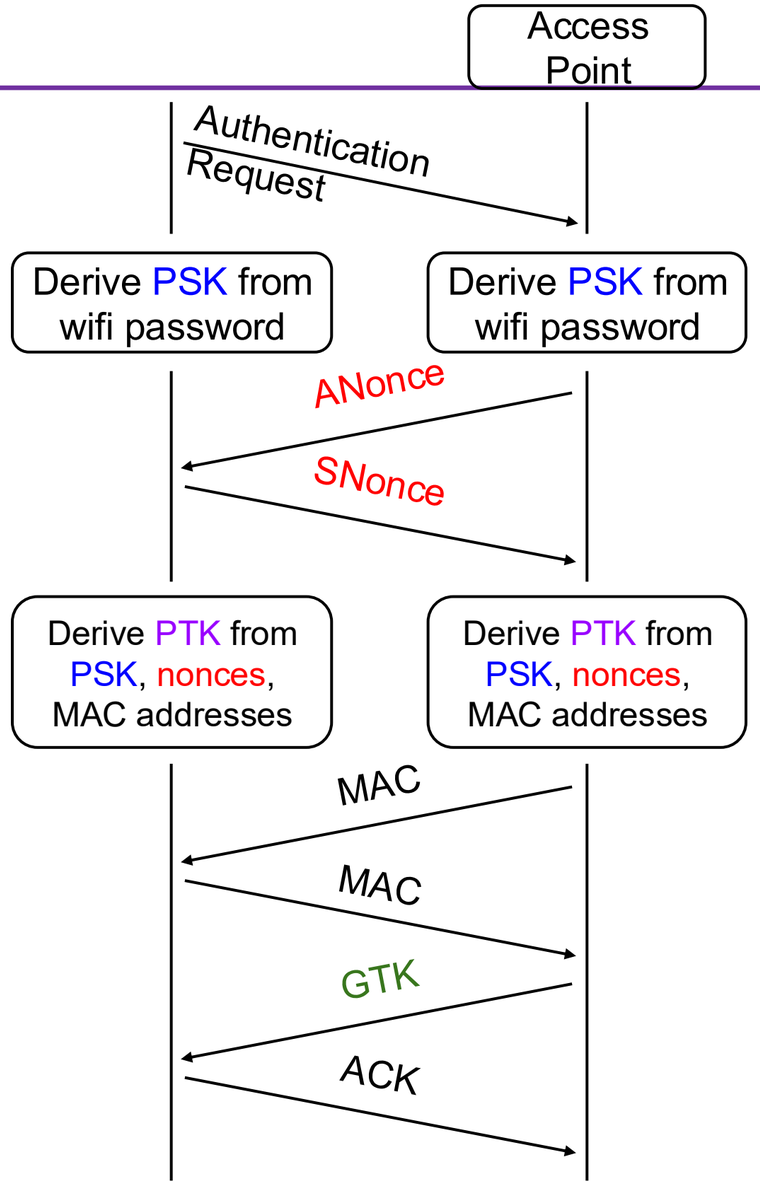
- **No forward secrecy**: An eavesdropper who records the values of *ANonce* and *SNonce* can derive the key if they later learn the password or *PSK*
  - Compare to Diffie-Hellman: An eavesdropper can't learn the key even if the record g^a and g^b and later compromise Alice's computer



Client: WPA-PSK Attacks

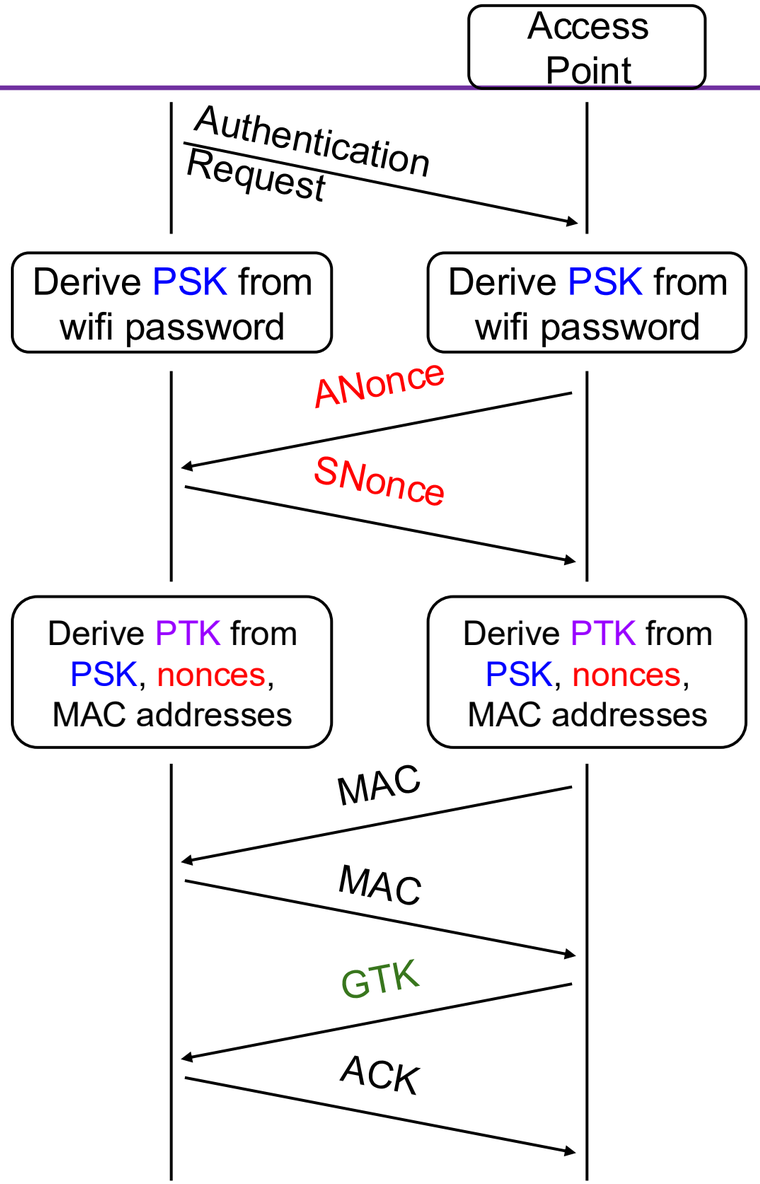
- **No forward secrecy**: Let's say the attacker records all the traffics between you and the coffee house (ANonce, Snounce, encrypted traffic). Even he/she does not know the wifi pasword now. However, as soon as the attacker get the wifi passworkd, then he/she can decrypt the traffic

$$\text{PTK} = \text{PRF}(\text{PSK} + \text{ANonce} + \text{SNonce} + \text{AP_MAC} + \text{Client_MAC})$$



Client: WPA-PSK Attacks

- **DHE: Have forward secrecy**
- An eavesdropper can't learn the key even if the record g^a and g^b and later compromise Alice's computer
  - That's because a and b has been cleared
  - The encryption key used is $K = g^{ab} \pmod{p}$



WPA-Enterprise



- **Core issue: Every client starts with the same *PSK* to derive the *PTK***
  - Fix: Have each user use their own username and password, instead
    - This is the model that AirBears2 and eduroam use!
- **Instead of using a PSK, use a key generated randomly by an authentication server**
  - For your client to trust the authentication server, you accept a digital certificate
  - Form a secure channel to the authentication server, which lets you enter your username and password
  - If the username and password are correct, the authentication server sends a one-time key to use instead of a PSK to both the client and the AP (also over a secure channel)
- **The rest of the handshake proceeds normally**

WPA-Enterprise Attacks



- **WPA Enterprise defends against the previous attacks**
  - **Rogue AP attack**: The APs must authenticate themselves to the server, which the attacker can't do
  - **Brute-force attack**: The generated PSK replacement is long and random, too long to brute-force
  - **No forward secrecy**: The generated PSK replacement is used once and then discarded, so no information is retained that allows the PTK to be recovered later
- **However, it is still vulnerable to attacks such as ARP or DHCP spoofing**

Summary



- **Classes of attackers:**
  - Off-path: Can't see, modify, or drop packets
  - On-path: Can see packets, but can't modify or drop packets
  - MITM: Can see, modify, and drop packets
- **ARP: A protocol to translate local IP addresses to MAC addresses**
  - Ask everyone on the network, "Who has the IP 1.2.3.4?"
  - Attack: The attacker can respond instead of the true device with 1.2.3.4, and packets will get routed to the attacker!
  - Defense: Switches
  - Defense: Rely on higher layers
- **DHCP: A protocol for a new client to receive a network configuration**
  - Ask everyone on the network, "What is the network configuration to use?"
  - Attack: The attacker can respond with a malicious configuration
  - Defense: Rely on higher layers

Summary



- **WPA: A protocol to encrypt Wi-Fi connections at layer 2**
  - Messages between the client and the AP are encrypted with keys
  - Handshake uses cryptographic MACs to verify that both parties have the same PSK and nonces
  - WPA-PSK: Use a password to derive a PSK, which is used in a handshake to arrive at a key
    - Attack: Attacker can pretend to be an AP
    - Attack: Brute-force the password after recording a handshake
    - Vulnerability: No forward secrecy
  - WPA-Enterprise: Use a third party to provide a one-time “replacement PSK,” used in the same handshake
    - Solves the attacks on WPA-PSK