Network Security: Attacks

CS 161: Computer Security

Prof. Vern Paxson

TAs: Paul Bramsen, Apoorva Dornadula,
David Fifield, Mia Gil Epner, David Hahn, Warren He,
Grant Ho, Frank Li, Nathan Malkin, Mitar Milutinovic,
Rishabh Poddar, Rebecca Portnoff, Nate Wang

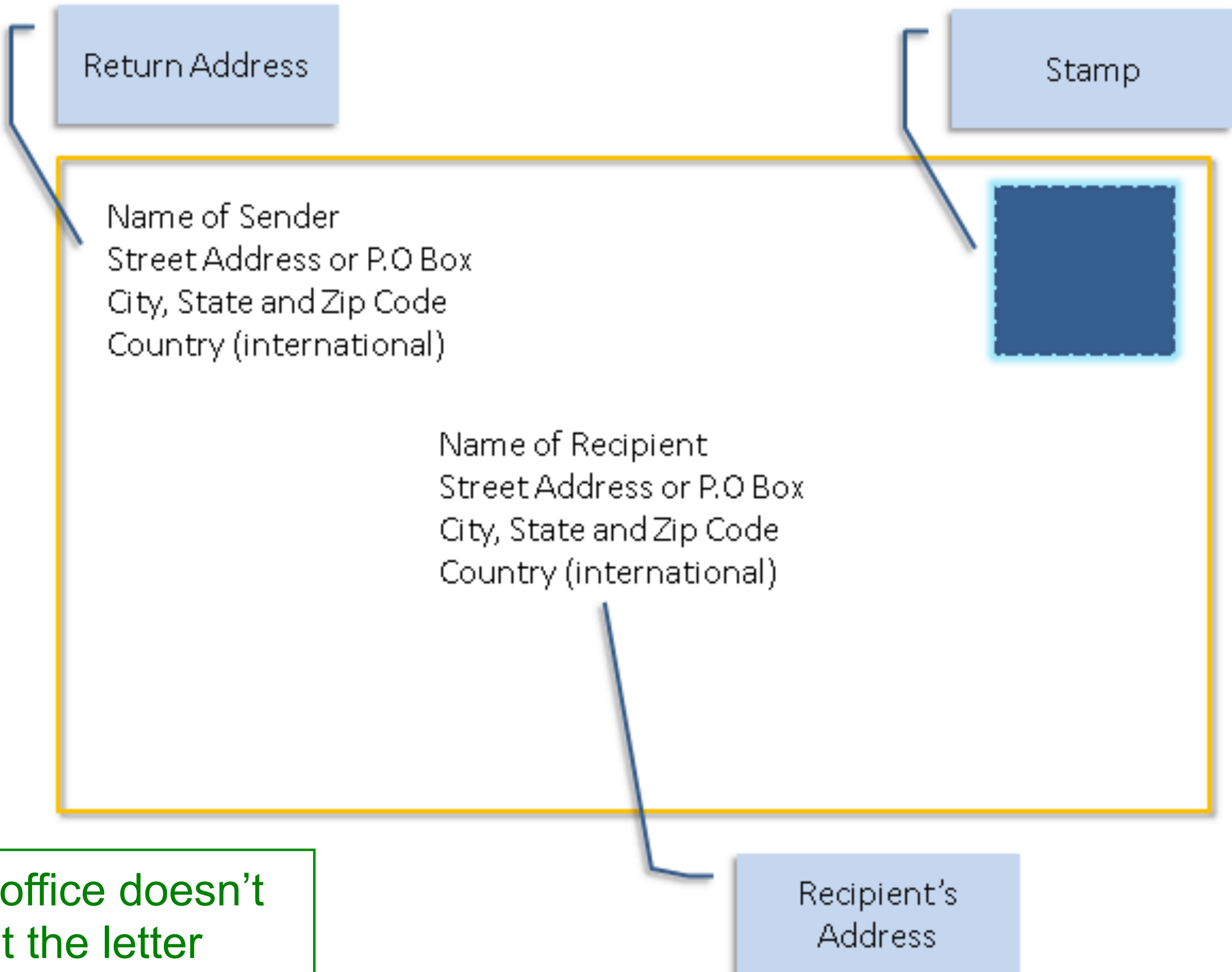
http://inst.eecs.berkeley.edu/~cs161/

March 9, 2017

IP Packet Header (Continued)

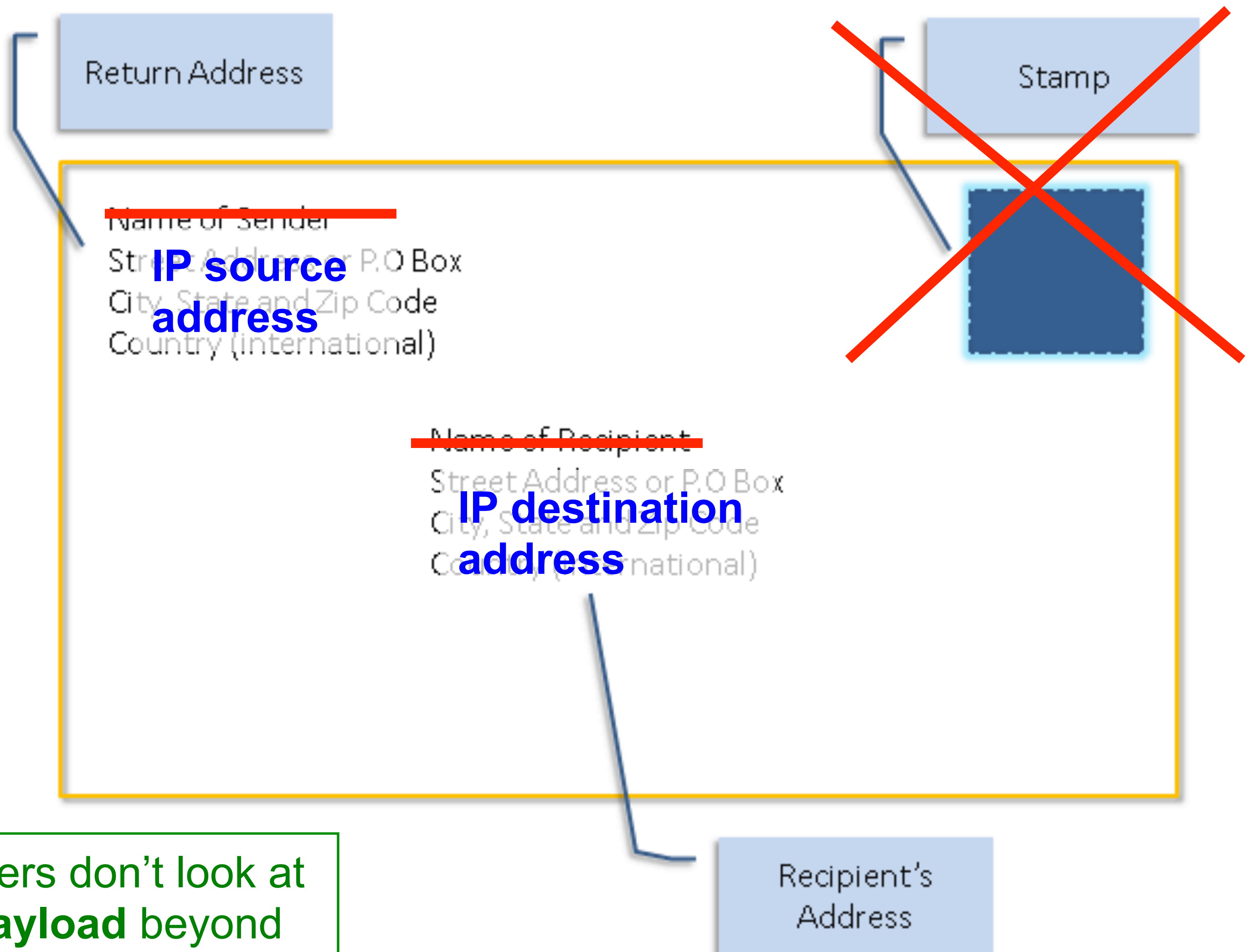
- Two IP addresses
  - Source IP address (32 bits in main IP version)
  - Destination IP address (32 bits, likewise)
- Destination address
  - Unique identifier/locator for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier/locator for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send reply back to source

Postal Envelopes:



(Post office doesn't look at the letter inside the envelope)

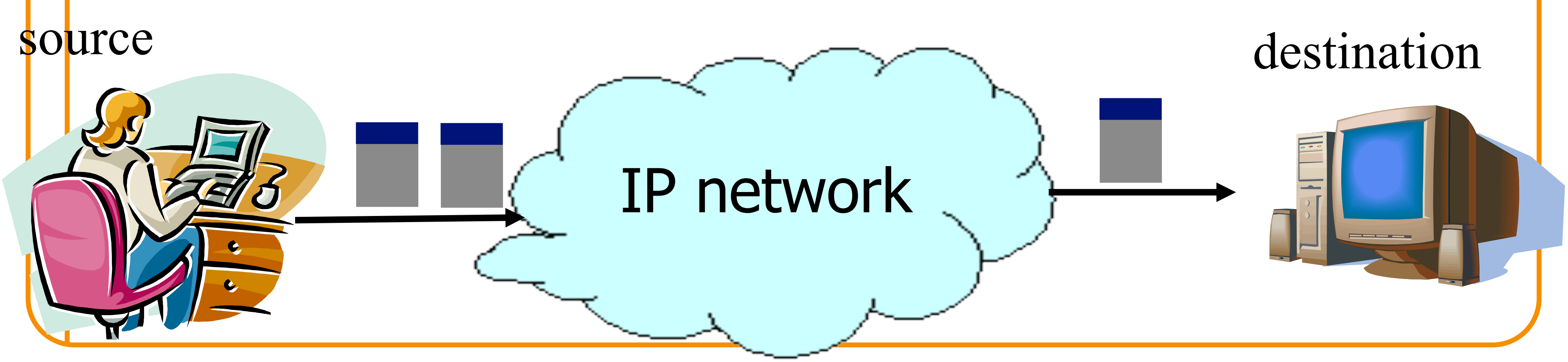
Analogy of IP to Postal Envelopes:



(Routers don't look at the **payload** beyond the IP header)

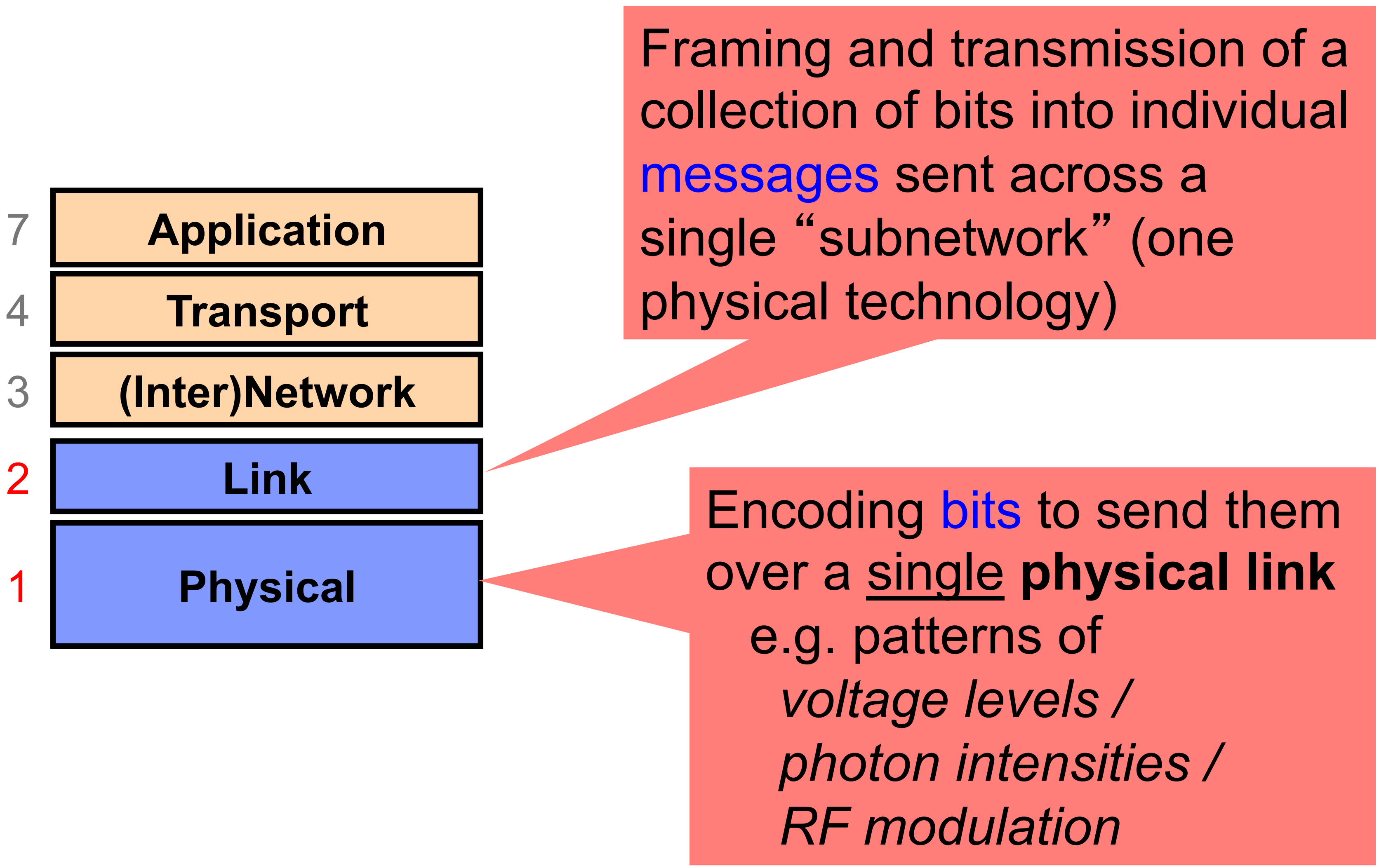
IP: "*Best Effort*" Packet Delivery

- Routers inspect destination address, locate "next hop" in forwarding table
  - Address = ~unique identifier/locator for the receiving host
- Only provides a "*I'll give it a try*" delivery service:
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



Threats Due to the Lower Layers

Layers 1 & 2: General Threats?



Physical/Link-Layer Threats: *Eavesdropping*

- Also termed *sniffing*
- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), get it for "free"
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - `tcpdump` (low-level ASCII printout)

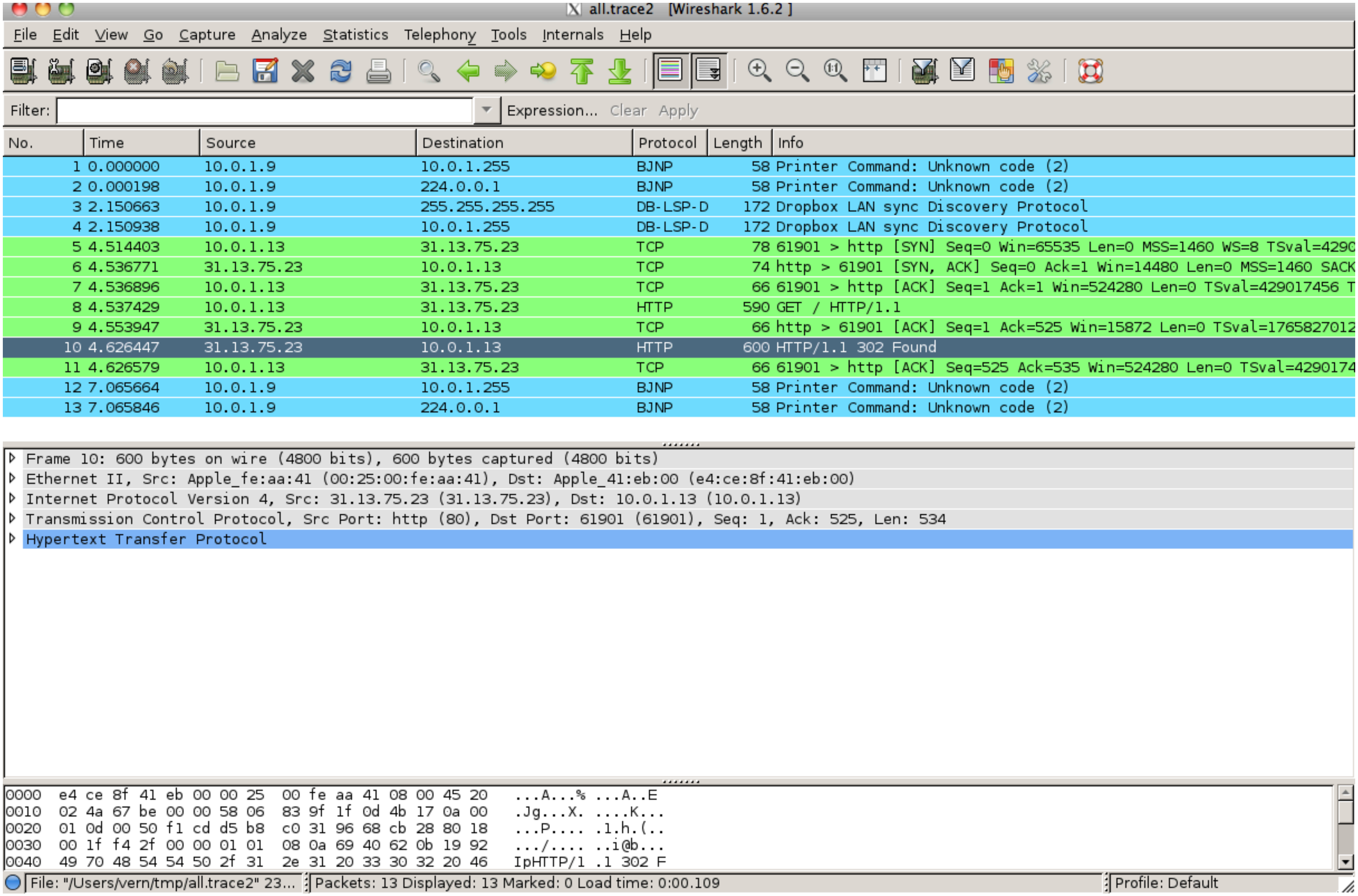
TCPDUMP: Packet Capture & ASCII Dumper

```
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
```

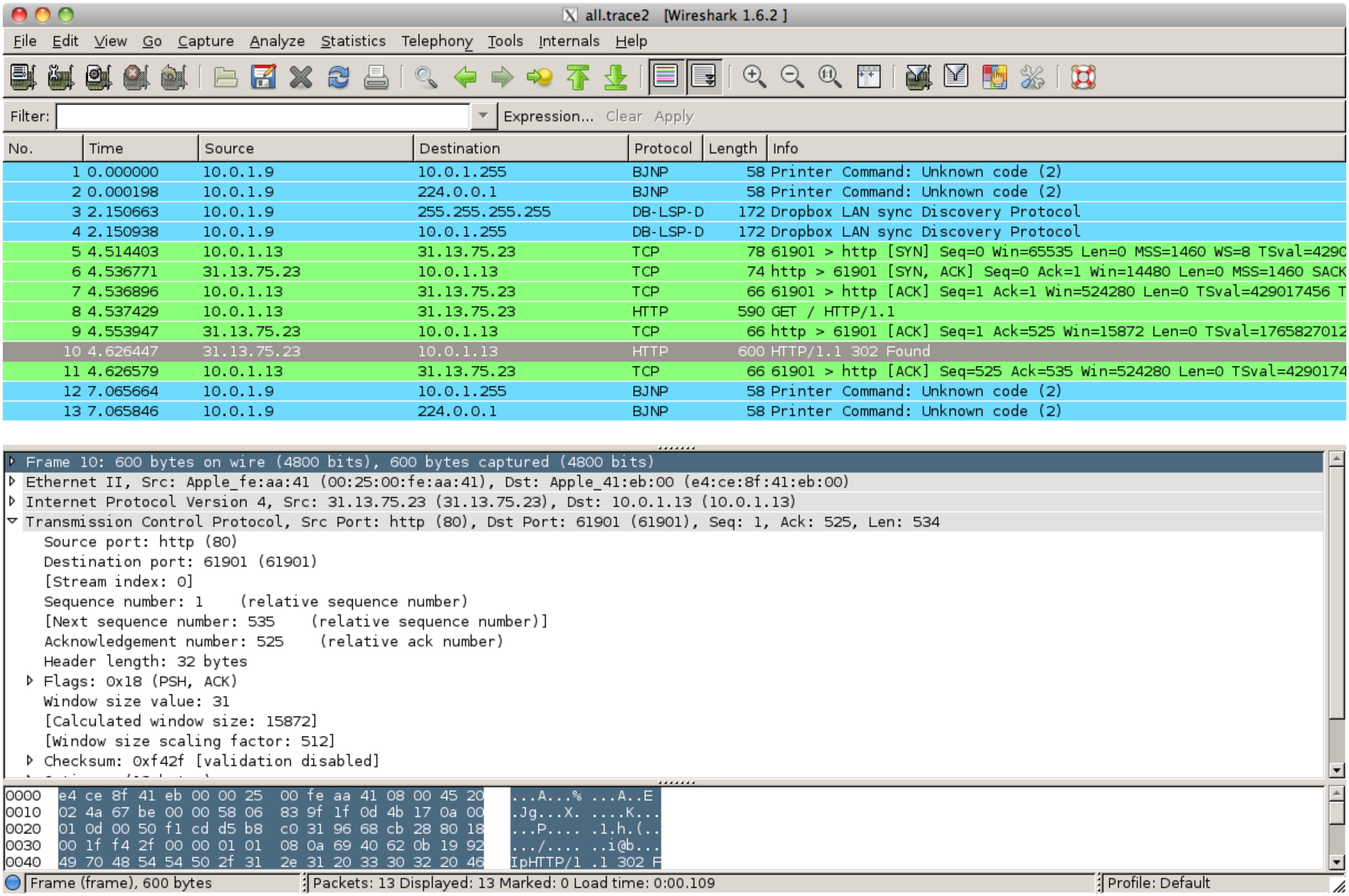
Physical/Link-Layer Threats: *Eavesdropping*

- Also termed *sniffing*
- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), get it for "free"
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - `tcpdump` (low-level ASCII printout)
    - Wireshark (GUI for displaying 800+ protocols)

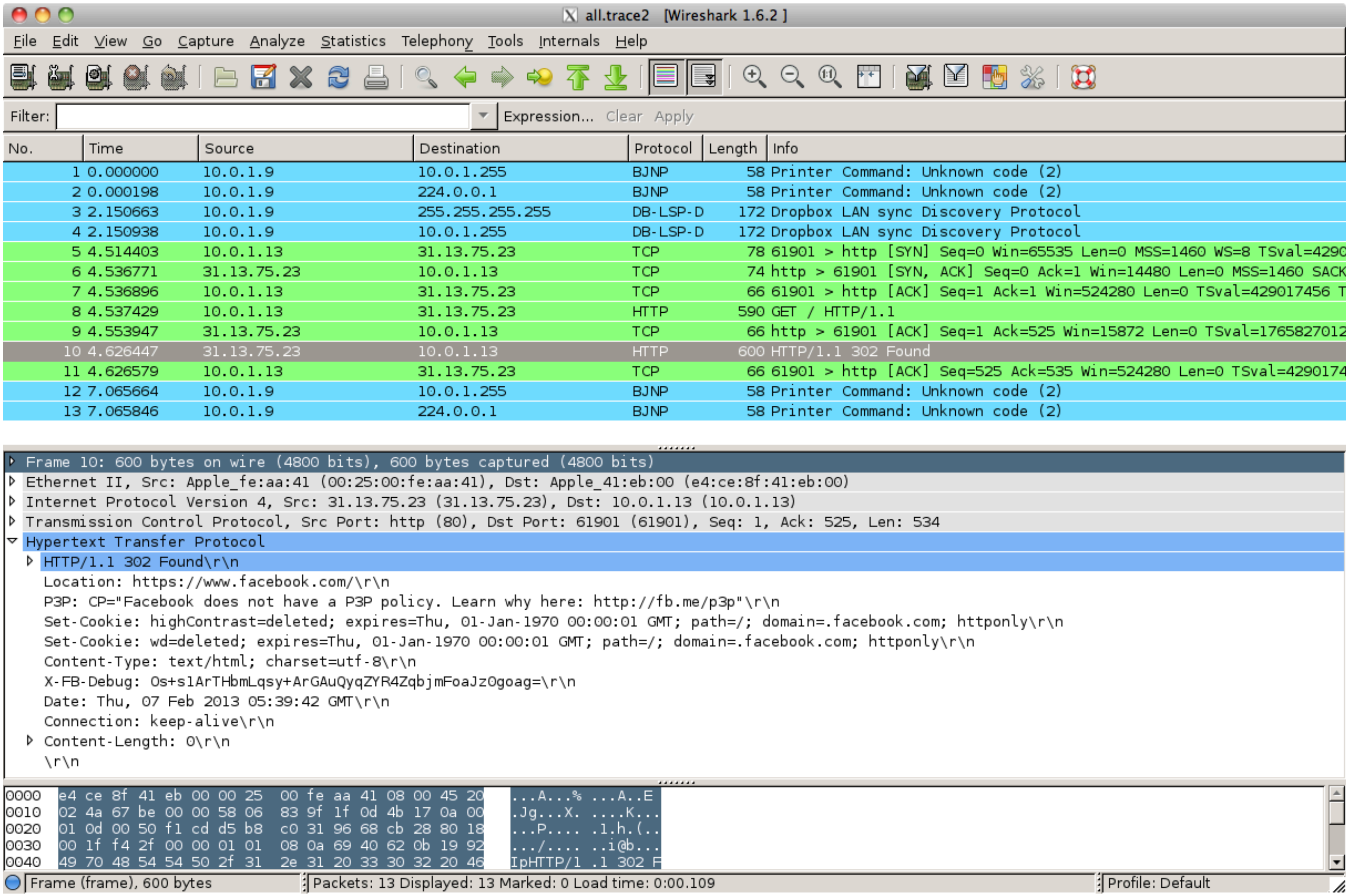
Wireshark: GUI for Packet Capture/Exam.



Wireshark: GUI for Packet Capture/Exam.



Wireshark: GUI for Packet Capture/Exam.



Physical/Link-Layer Threats: *Eavesdropping*

- Also termed *sniffing*
- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), get it for "free"
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - `tcpdump` (low-level ASCII printout)
    - Wireshark (GUI for displaying 800+ protocols)
    - Bro (scriptable real-time network analysis; see `bro.org`)
- For any technology, routers (and internal "switches") can look at / export traffic they forward
- You can also "tap" a link
  - Insert a device to mirror the physical signal

Operation Ivy Bells

By Matthew Carle
Military.com

At the beginning of the 1970's, divers from the specially-equipped submarine, USS Halibut (SSN 587), left their decompression chamber to start a bold and dangerous mission, code named "Ivy Bells".

The Regulus guided missile submarine, USS Halibut (SSN 587) which carried out Operation Ivy Bells.

In an effort to alter the balance of Cold War, these men scoured the ocean floor for a five-inch diameter cable carry secret Soviet communications between military bases.

The divers found the cable and installed a 20-foot long listening device on the cable. designed to attach to the cable without piercing the casing, the device recorded all communications that occurred. If the cable malfunctioned and the Soviets raised it for repair, the bug, by design, would fall to the bottom of the ocean. Each month Navy divers retrieved the recordings and installed a new set of tapes.

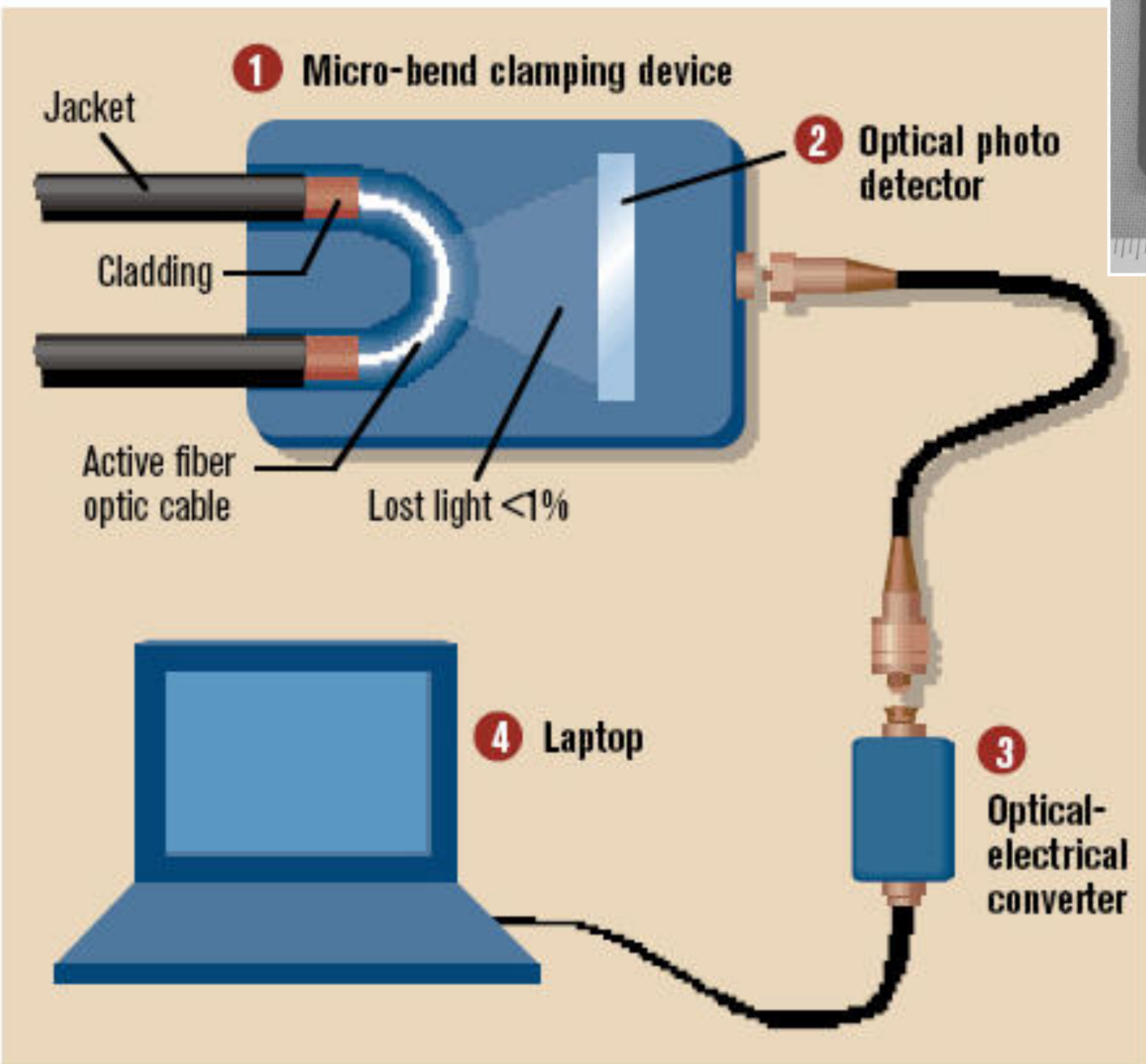
Upon their return to the United States, intelligence agents from the NSA analyzed the recordings and tried to decipher any encrypted information. The Soviets apparently were confident in the security of their communications lines, as a surprising amount of sensitive information traveled through the lines without encryption.

prison. The original tap that was discovered by the Soviets is now on exhibit at the KGB museum in Moscow.

Physical/Link-Layer Threats: *Eavesdropping*

- Also termed *sniffing*
- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), get it for "free"
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - `tcpdump` (low-level ASCII printout)
    - Wireshark (GUI for displaying 800+ protocols)
    - Bro (scriptable real-time network analysis)
- For any technology, routers (and internal "switches") can look at / export traffic they forward
- You can also "tap" a link
  - Insert a device to mirror the physical signal
  - Or: just steal it!

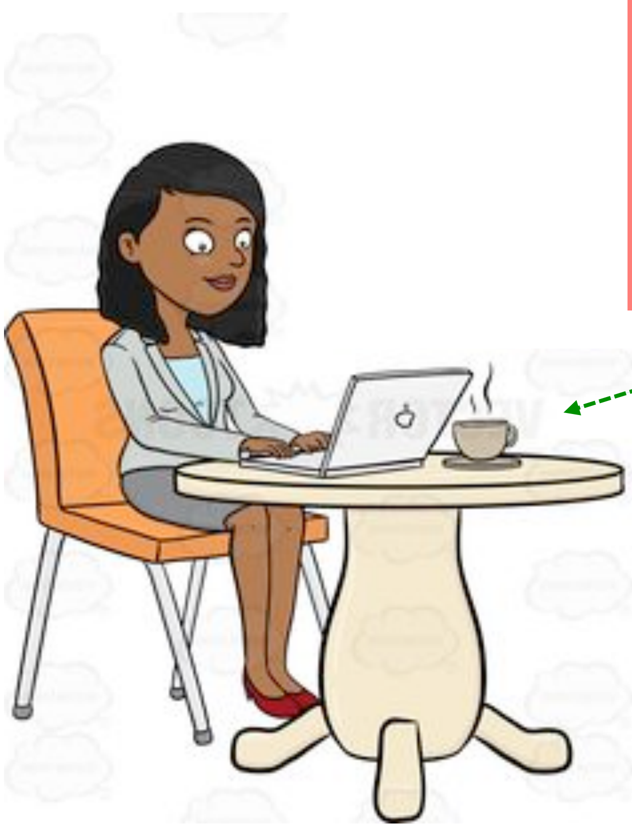
Stealing Photons



Protecting Against Eavesdropping in the Coffee Shop

Coffee Shop

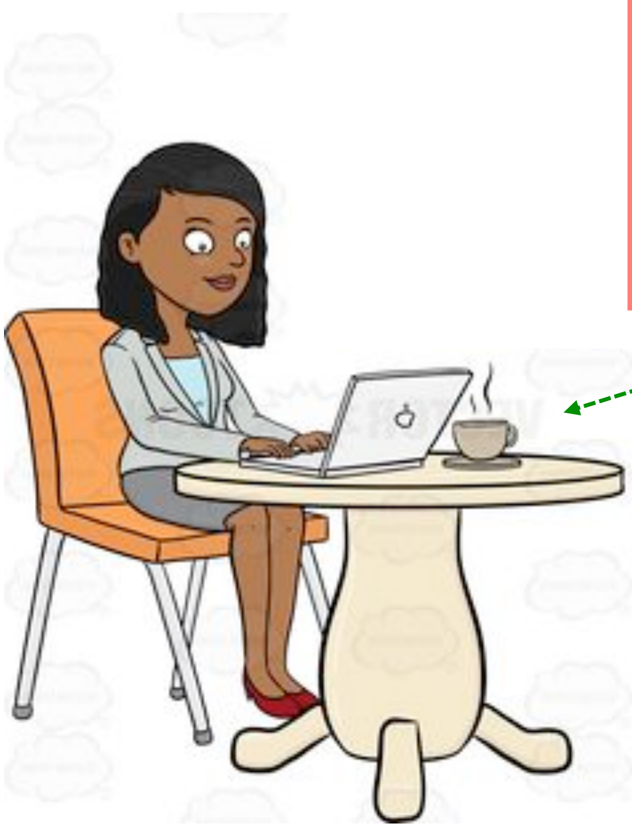
1. Join the wireless network



If either match up, your laptop joins the network. Optionally performs a cryptographic exchange.

Coffee Shop

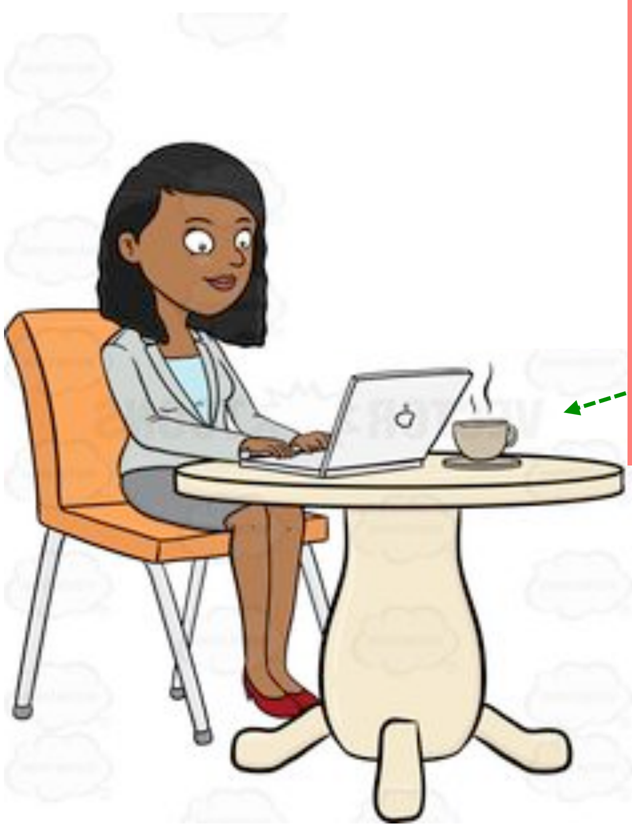
1. Join the wireless network



If either match up, your laptop joins the network. **Optionally performs a cryptographic exchange.**

Coffee Shop

1. Join the wireless network



If either match up, your laptop joins the network. **Optionally performs a cryptographic exchange.** Most commonly today, that is done using **WPA2.**

Coffee Shop

Password: `$secret!`

KeyCounter
(and other stuff)

KeyCounter
(and other stuff)

WPA2, common form ("Personal"; simplified)

SSID

Password: **`$secret!`**

KeyCounter
(and other stuff)

Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

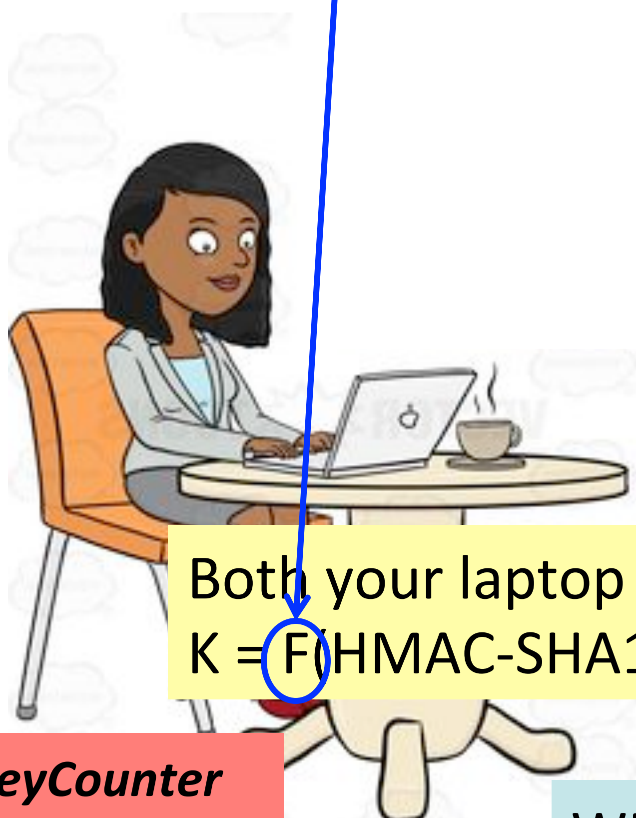
WPA2, common form (“Personal”; simplified)

Coffee Shop

This function

Password: `$secret!`

KeyCounter
(and other stuff)



Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

Coffee Shop

This function
computes this many iterations

Password: `$secret!`

KeyCounter
(and other stuff)

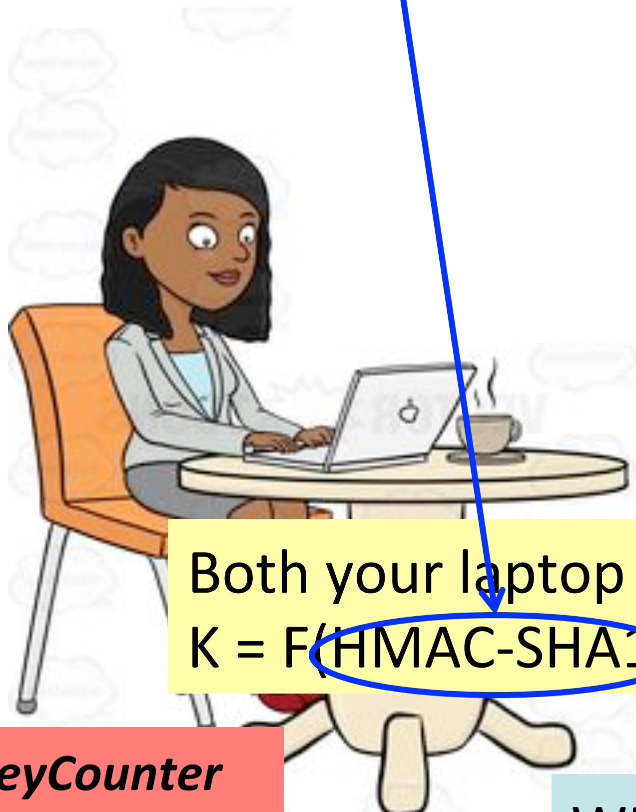
Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

Coffee Shop

This function
computes this many iterations
of this function



Password: `$secret!`

KeyCounter
(and other stuff)

Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

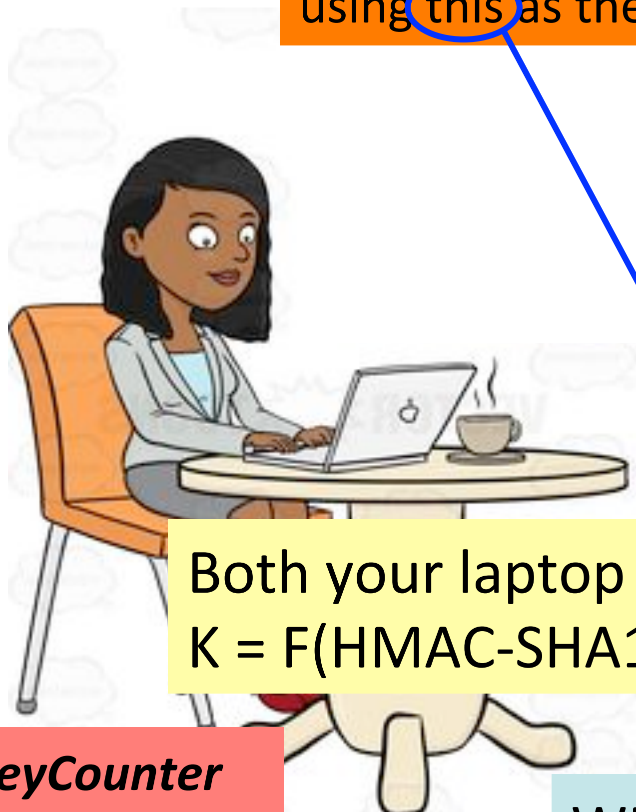
WPA2, common form (“Personal”; simplified)

Coffee Shop

This function
computes this many iterations
of this function
using this as the MAC key

Password: `$secret!`

KeyCounter
(and other stuff)



Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!”, “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

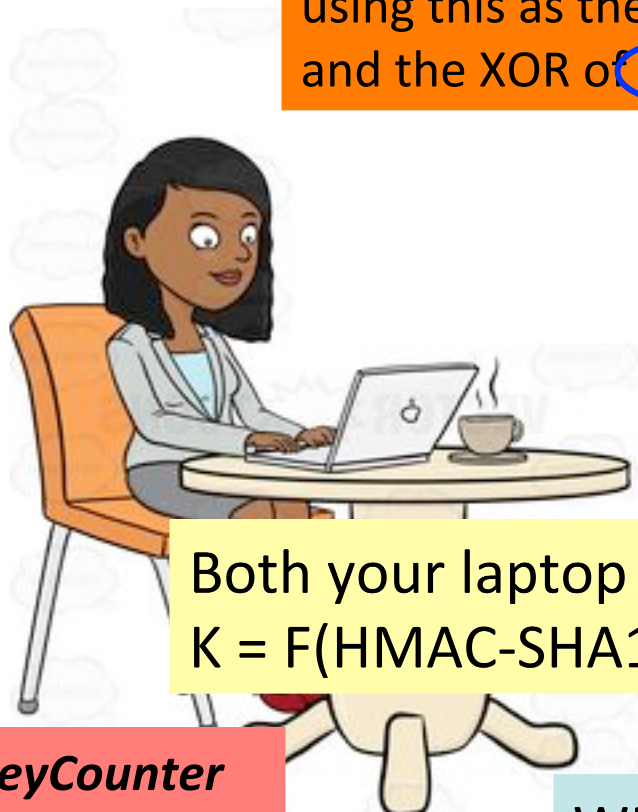
Coffee Shop

This function
computes this many iterations
of this function
using this as the MAC key
and the XOR of these as the initial input.

Password: `$secret!`

KeyCounter
(and other stuff)

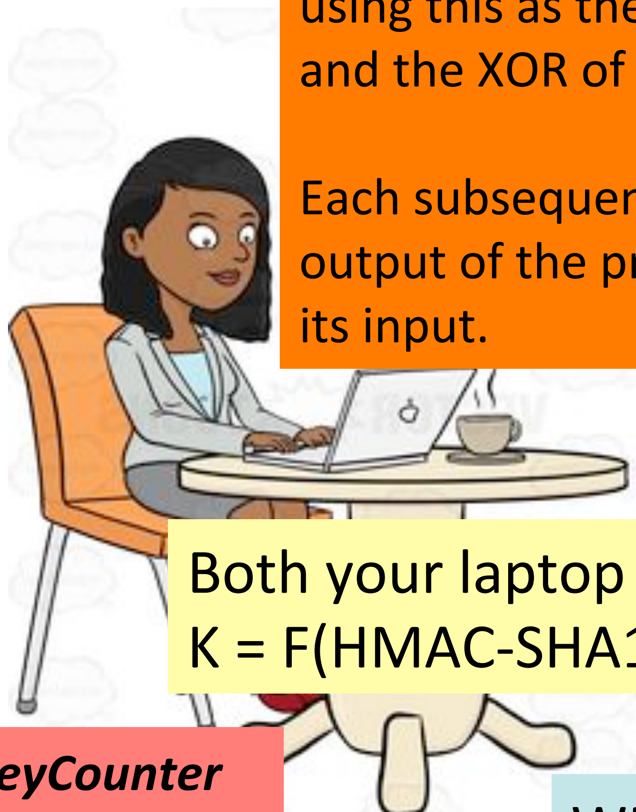
Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)



KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

Coffee Shop



This function
computes this many iterations
of this function
using this as the MAC key
and the XOR of these as the initial input.

Each subsequent iteration takes the output of the previous computation as its input.

Password: `$secret!`

KeyCounter
(and other stuff)

Both your laptop and the AP now compute:
K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

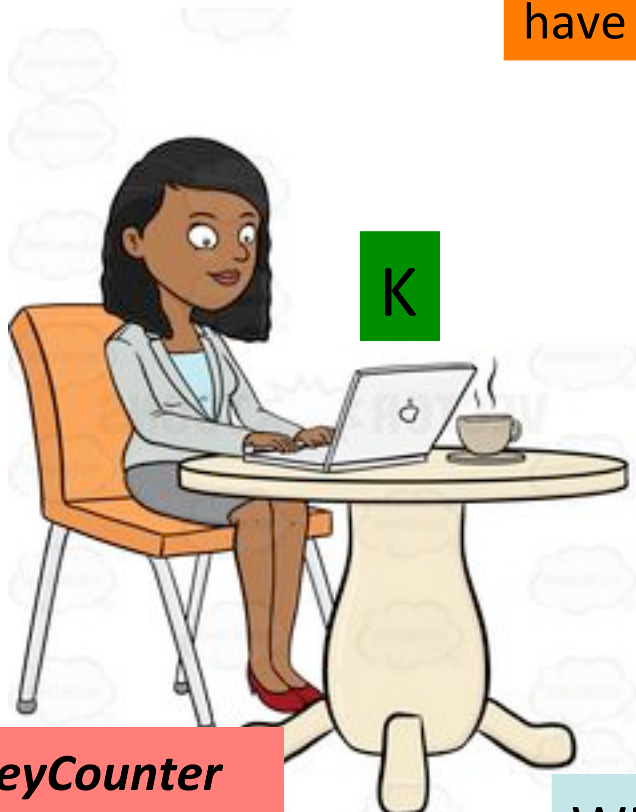
Coffee Shop

Now your laptop and the AP have *derived* a shared secret.

K

Password: `$secret!`

KeyCounter
(and other stuff)



K

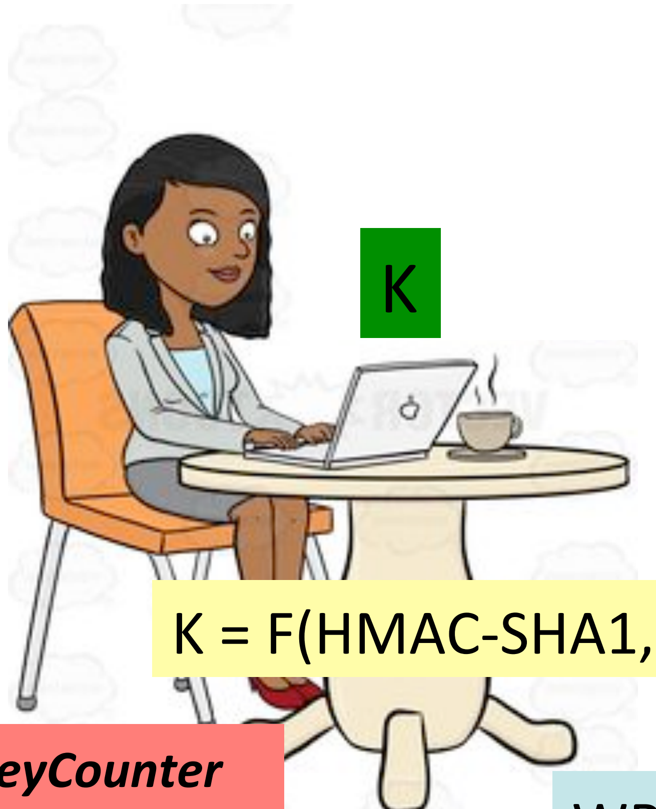
KeyCounter
(and other stuff)

WPA2, common form ("Personal"; simplified)


Eve

Coffee Shop

Eve attacks!



K

K

Password: `$Secret!`

KeyCounter
(and other stuff)

K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

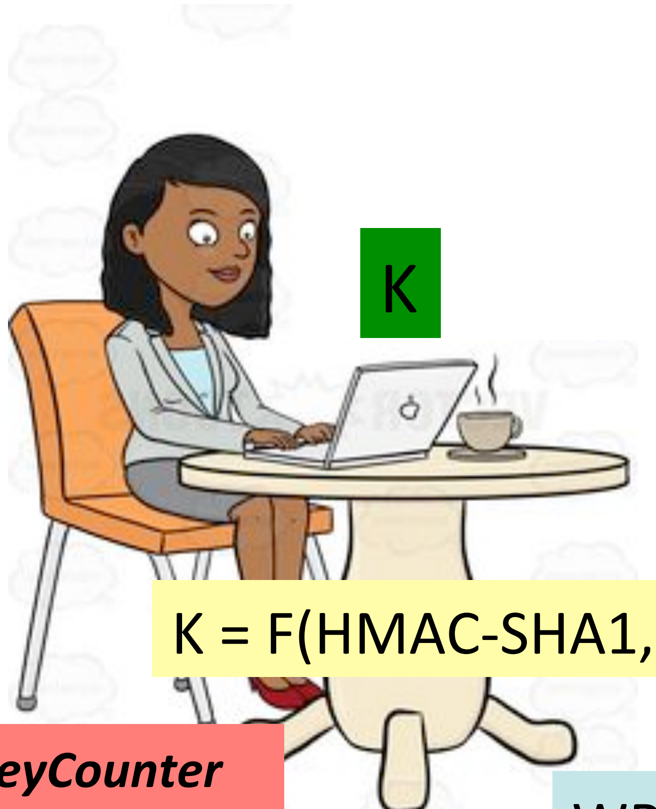
WPA2, common form (“Personal”; simplified)

Eve

Coffee Shop

Since the password is never exposed, if Eve doesn't know it, the best she can do is a **dictionary attack** to try to *guess* it.

K



K

Password: `$secret!`

KeyCounter
(and other stuff)

K = F(HMAC-SHA1, "$secret!", "ATT192", *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form ("Personal"; simplified)

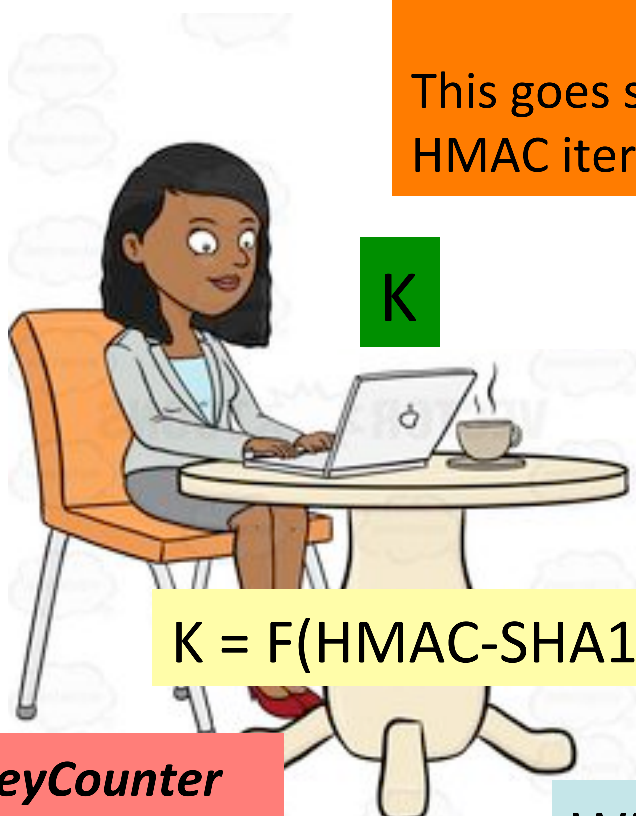
Eve

Since the password is never exposed, if Eve doesn't know it, the best she can do is a **dictionary attack** to try to *guess it*.

This goes slowly due to the 1000s of HMAC iterations.

K

K



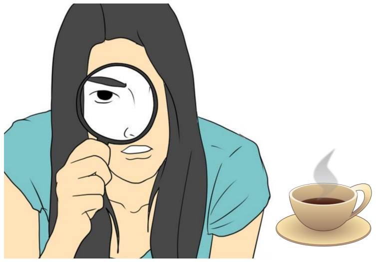
Password: `$secret!`

KeyCounter
(and other stuff)

K = F(HMAC-SHA1, “\$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

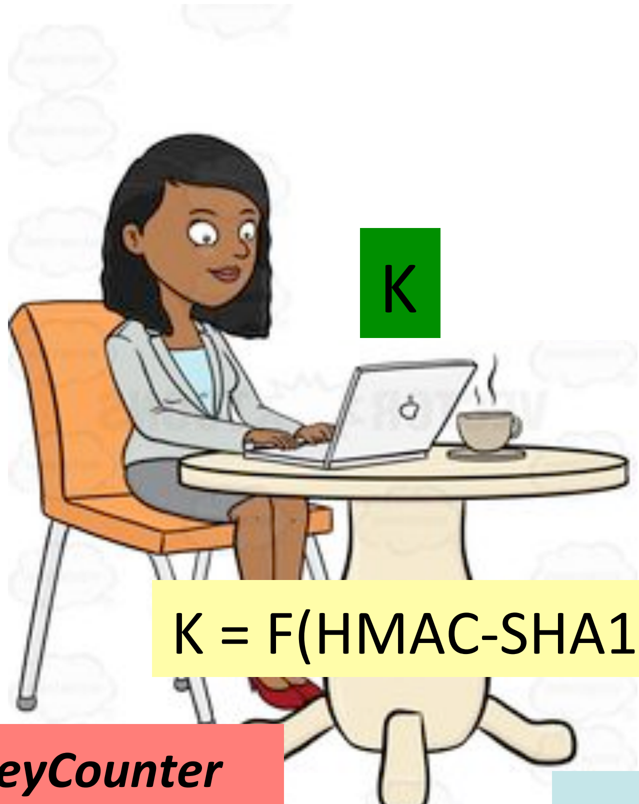


Eve

Coffee Shop

BUT: if Eve ponies up $2.25 for a cup of coffee and gets the password to the local net …

K



K

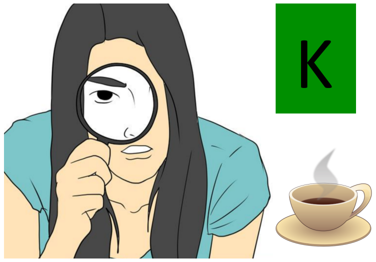
Password: `$secret!`

KeyCounter
(and other stuff)

K = F(HMAC-SHA1, “$secret!", “ATT192”, *KeyCounter*, 4096)

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)



K

Eve

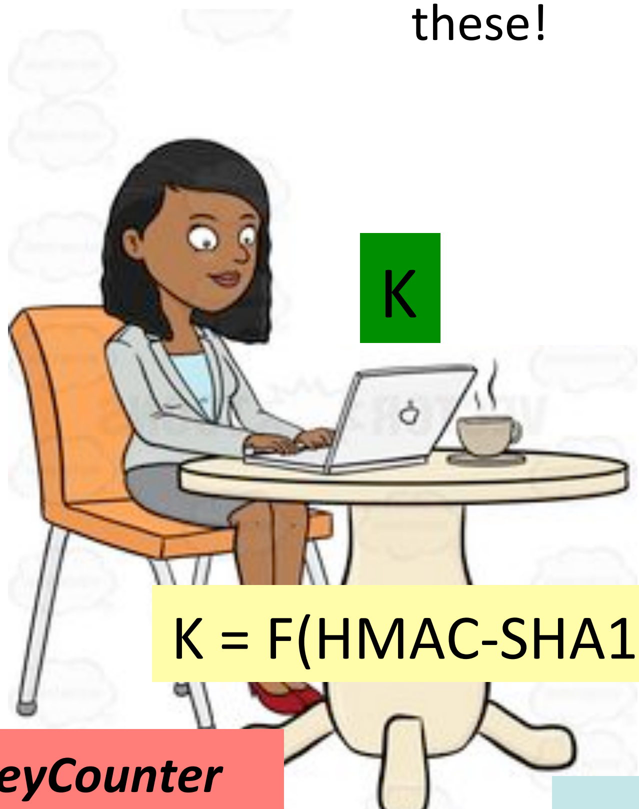
Coffee Shop

BUT: if Eve ponies up $2.25 for a cup of coffee and gets the password to the local net … then she knows both of these!

K

Password: `$secret!`

KeyCounter
(and other stuff)



K

K = F(HMAC-SHA1, "$secret!", "ATT192", *KeyCounter*, 4096)

KeyCounter
(and other stuff)

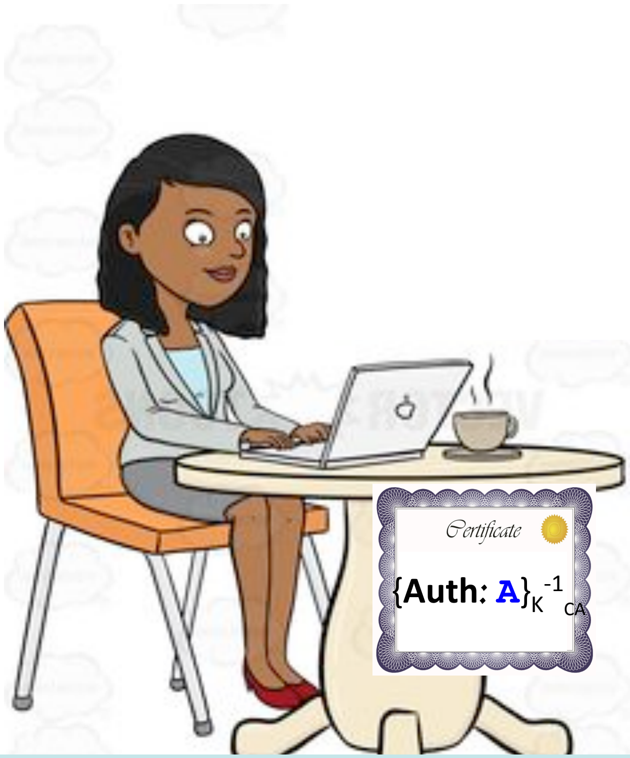
WPA2, common form ("Personal"; simplified)

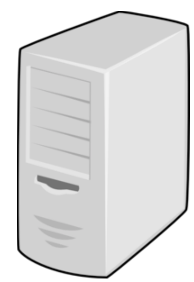
Coffee Shop

WPA2, actually-secure-but-inconvenient form(“Enterprise”; simplified)

Coffee Shop

Your laptop is *preconfigured* with a cert for an **Authentication Server**.





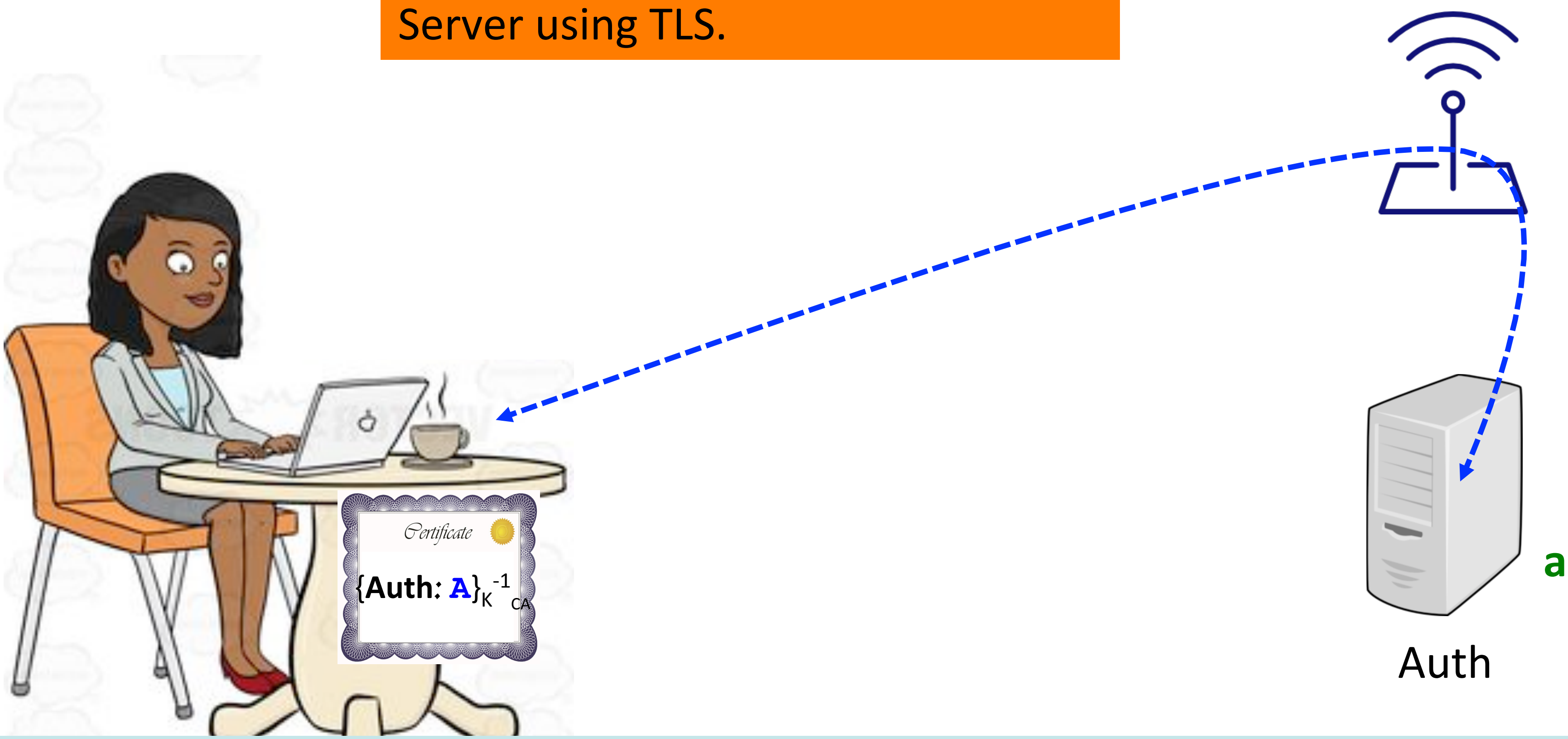
a

Auth

WPA2, actually-secure-but-inconvenient form("Enterprise"; simplified)

Coffee Shop

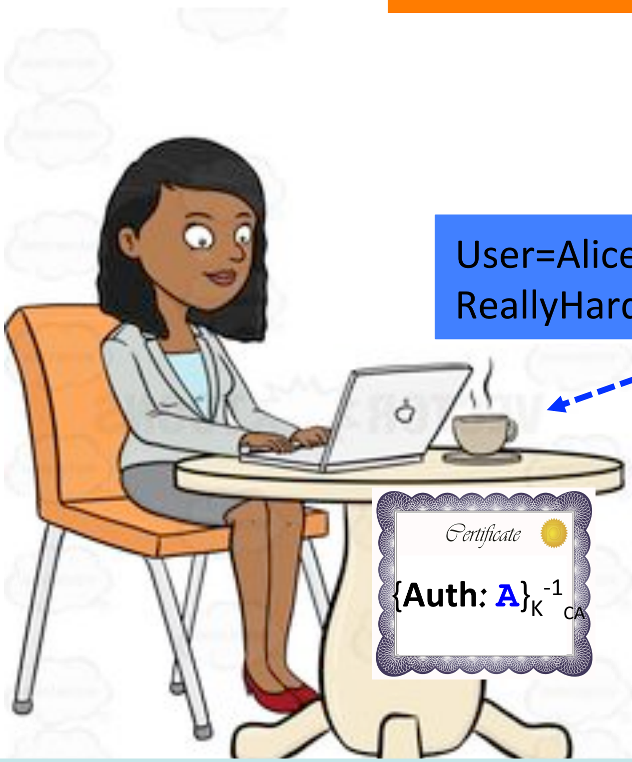
You establish a secure connection via the AP to the Authentication Server using TLS.



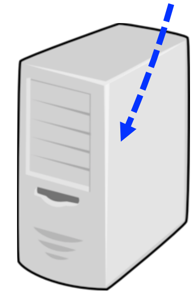
WPA2, actually-secure-but-inconvenient form("Enterprise"; simplified)

Coffee Shop

You then transmit your authentication info (username/password, or your own cert) to the server



User=Alice, Password=
ReallyHard2Gue$$



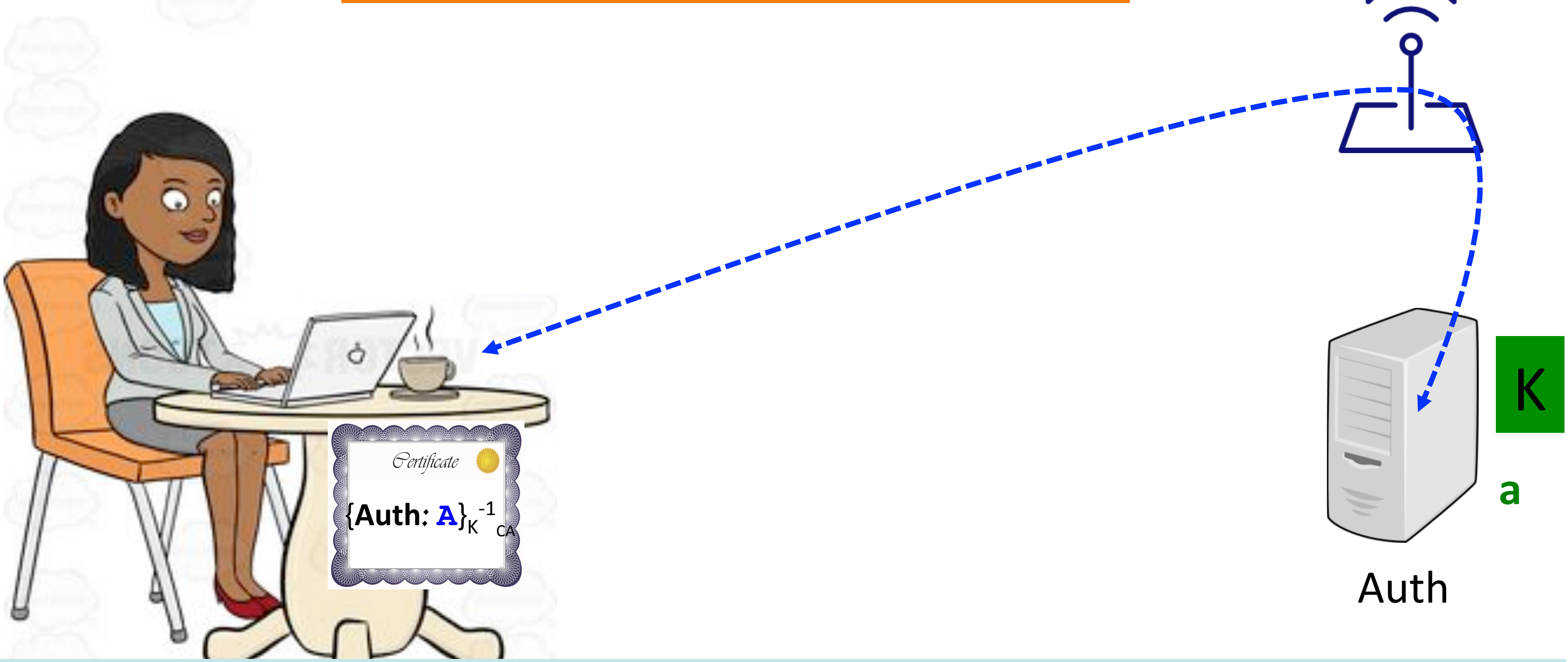
a

Auth

WPA2, actually-secure-but-inconvenient form("Enterprise"; simplified)

Coffee Shop

The Authentication Server creates a random secret key and sends it to both your laptop and the AP.



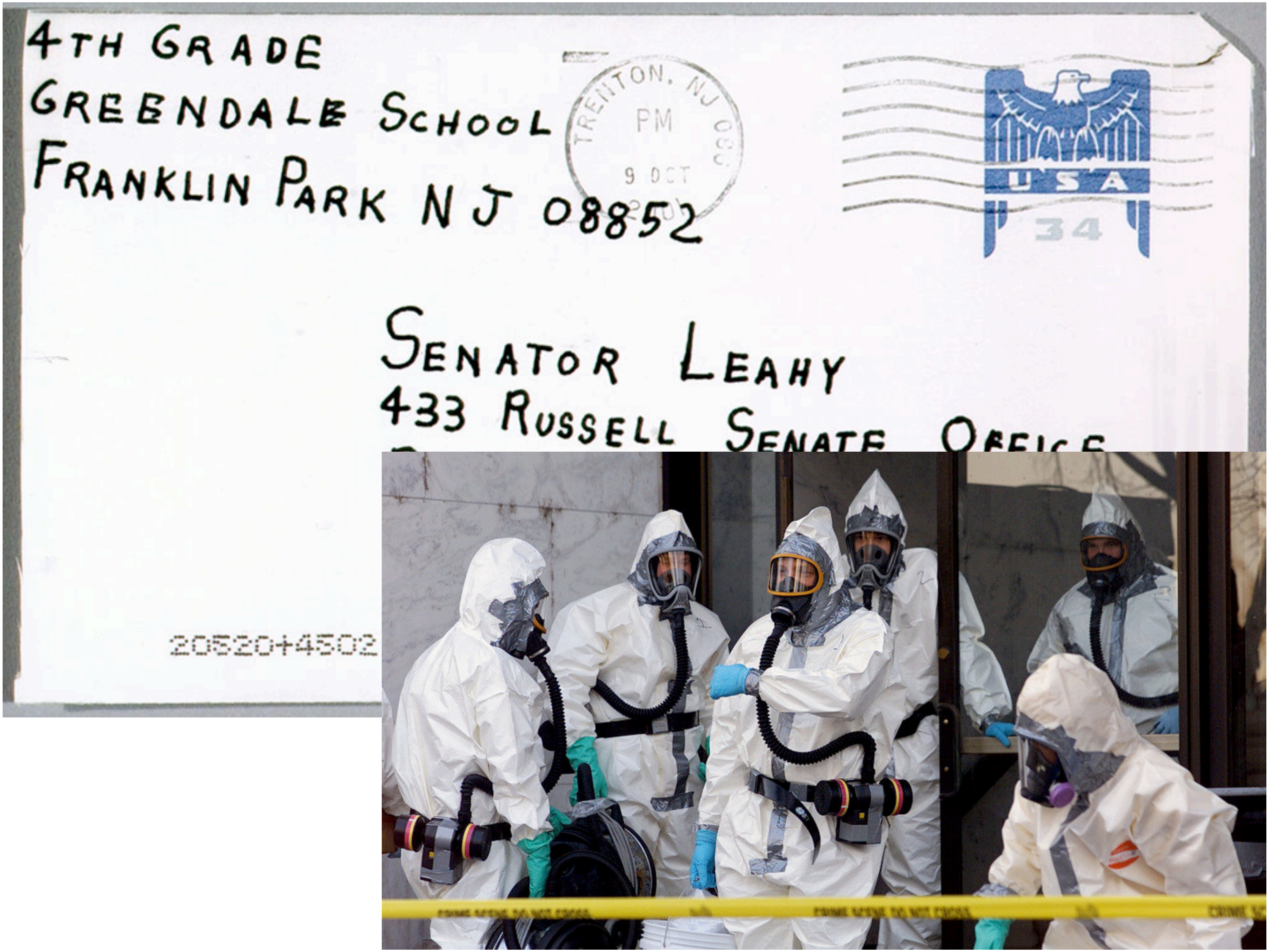
WPA2, actually-secure-but-inconvenient form(“Enterprise”; simplified)

5 Minute Break

Questions Before We Proceed?

Physical/Link-Layer Threats: *Spoofing*

- With physical access to a subnetwork, attacker can create any message they like
  - When with a bogus source address: *spoofing*


4TH GRADE
GREENDALE SCHOOL
FRANKLIN PARK NJ 08852
TRENTON, NJ
PM
9 OCT
USA
34
SENATOR LEAHY
433 RUSSELL SENATE OFFICE
20520+4502

Physical/Link-Layer Threats: *Spoofing*

- With physical access to a subnetwork, attacker can create any message they like
  - When with a bogus source address: *spoofing*
- When using a typical computer, may require root/administrator to have full freedom
- Particularly powerful when combined with *eavesdropping*
  - Because attacker can understand exact state of victim's communication and craft their spoofed traffic to match it
  - Spoofing w/o eavesdropping = "*blind spoofing*"

Spoofing Considerations

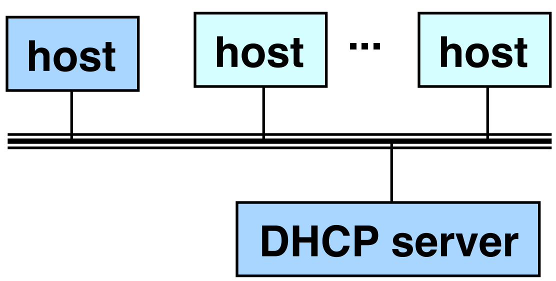
- "On path" attackers can see victim's traffic ⇒ spoofing is easy
- "Off path" attackers can't see victim's traffic
  - They have to resort to blind spoofing
  - Often must guess/infer header values to succeed
    - We care about the work factor: how hard is this
  - Sometimes they can just **brute force**
    - E.g., 16-bit value: just try all 65,536 possibilities!
- When we say an attacker "can spoof", we usually mean "w/ feasible chance of achieving their goal"

Coffee Shop

2. Configure your connection

Internet Bootstrapping: DHCP

- New host doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know *who to ask* for an IP address
  - So, host doesn't know what destination address to use
- (Note, host does have a separate WiFi address)
- Solution: *shout* to "**discover**" server that can help
  - Broadcast a server-discovery message (layer 2)
  - Server(s) sends a reply offering an address



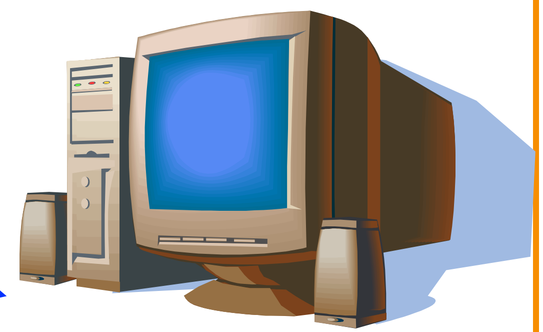
DHCP = Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol

new
client

DHCP discover
(broadcast)

DHCP offer



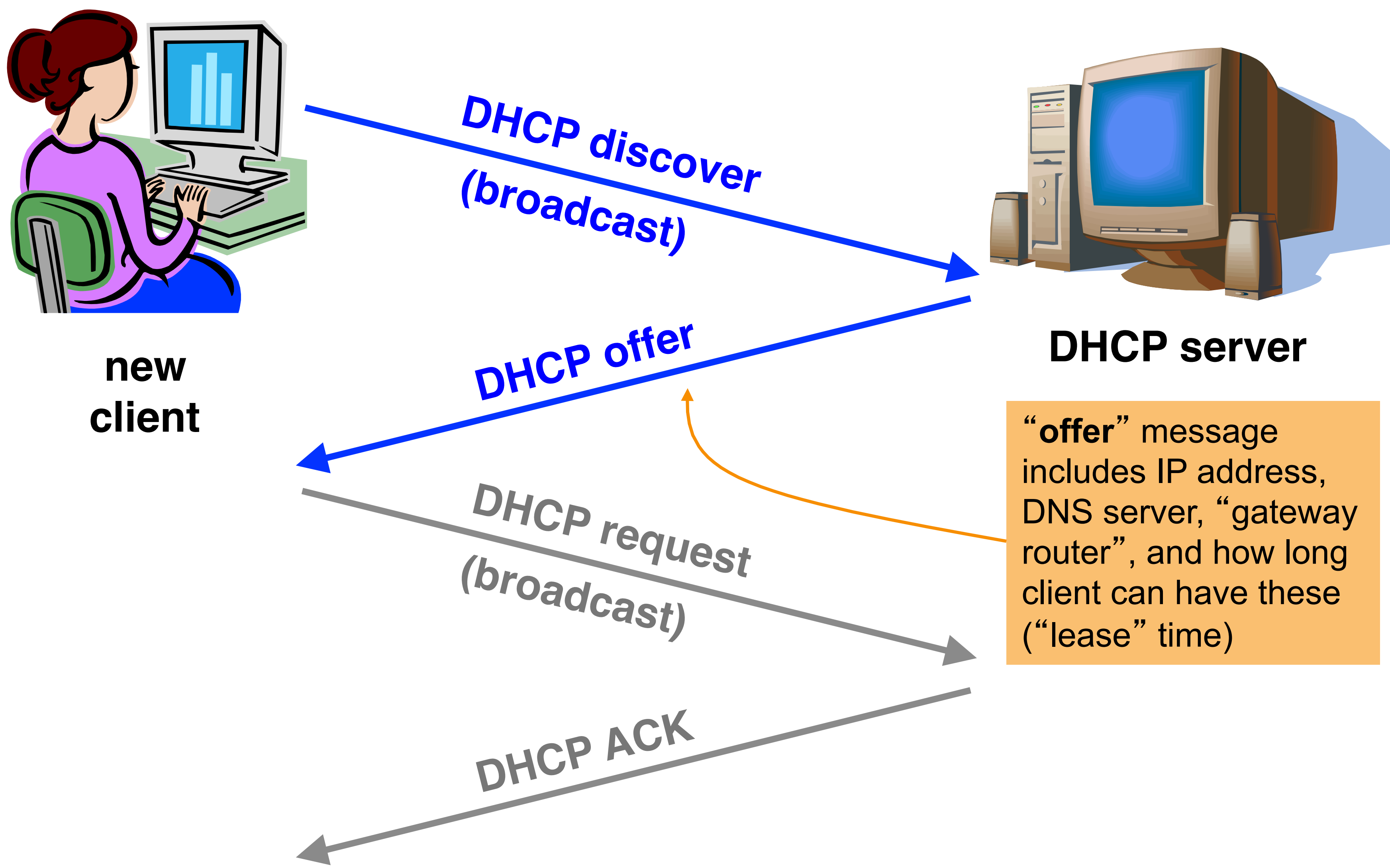
DHCP server

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

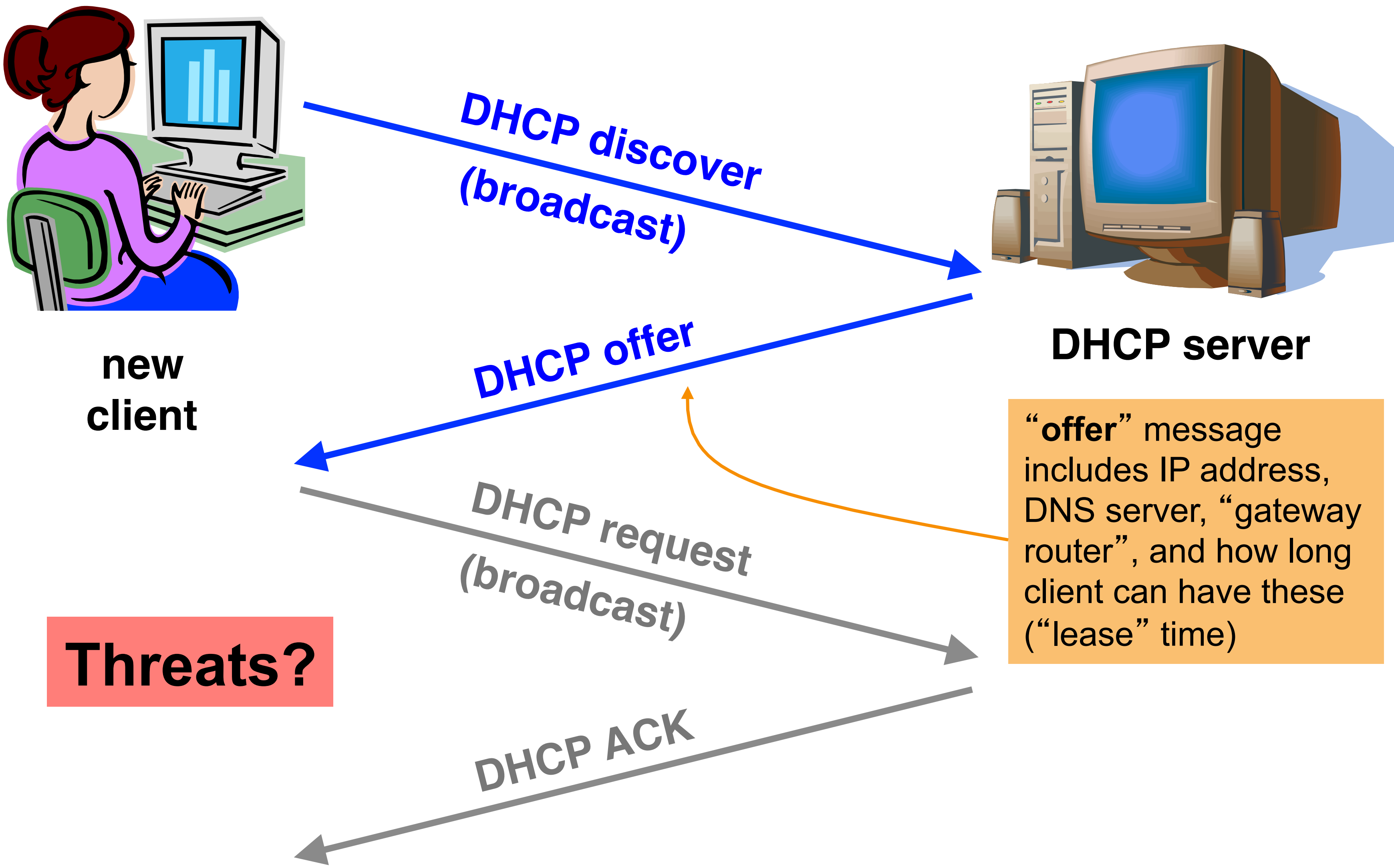
DNS server = system used by client to map hostnames like `gmail.com` to IP addresses like `74.125.224.149`

Gateway router = router that client uses as the first hop for all of its Internet traffic to remote hosts

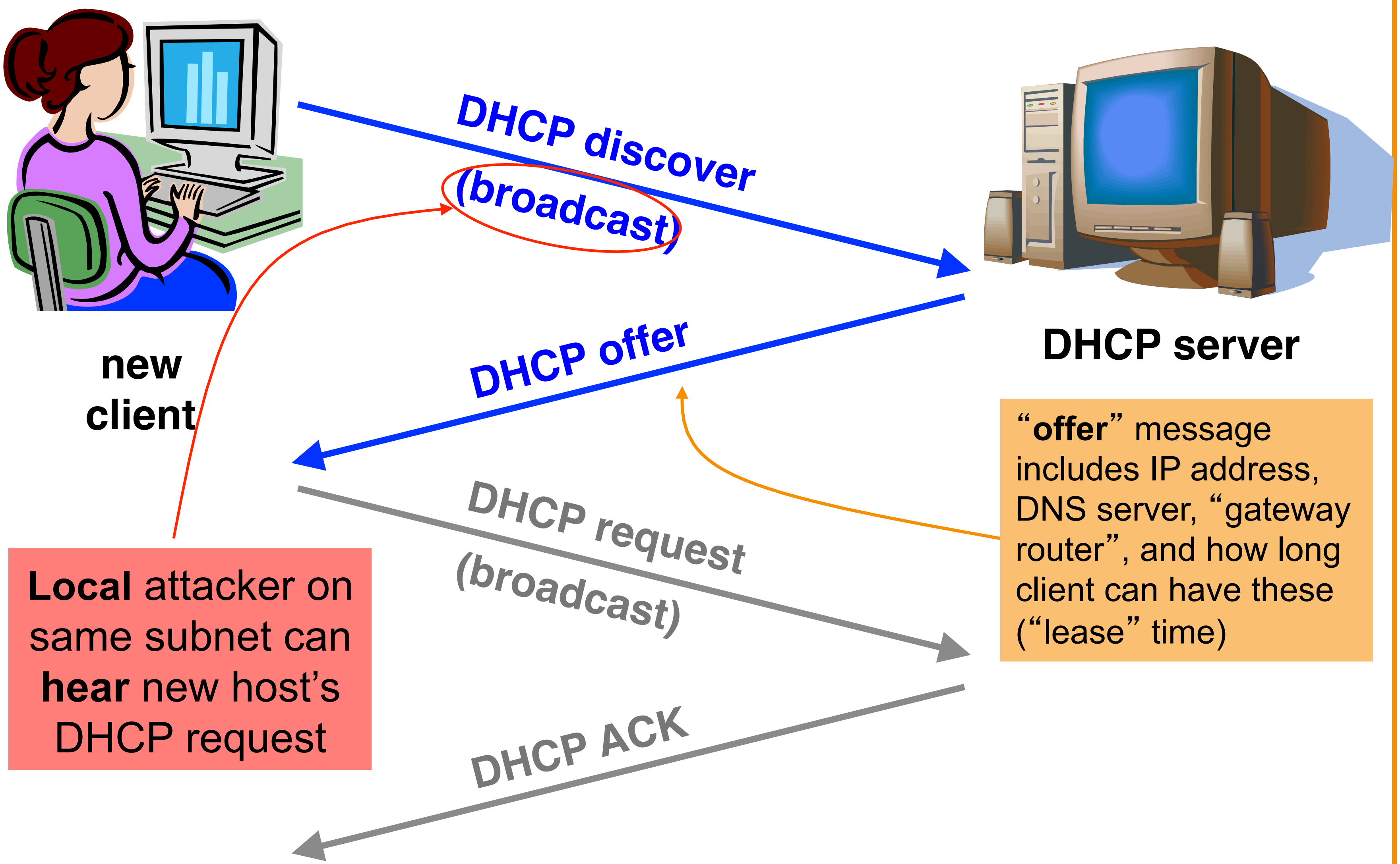
Dynamic Host Configuration Protocol



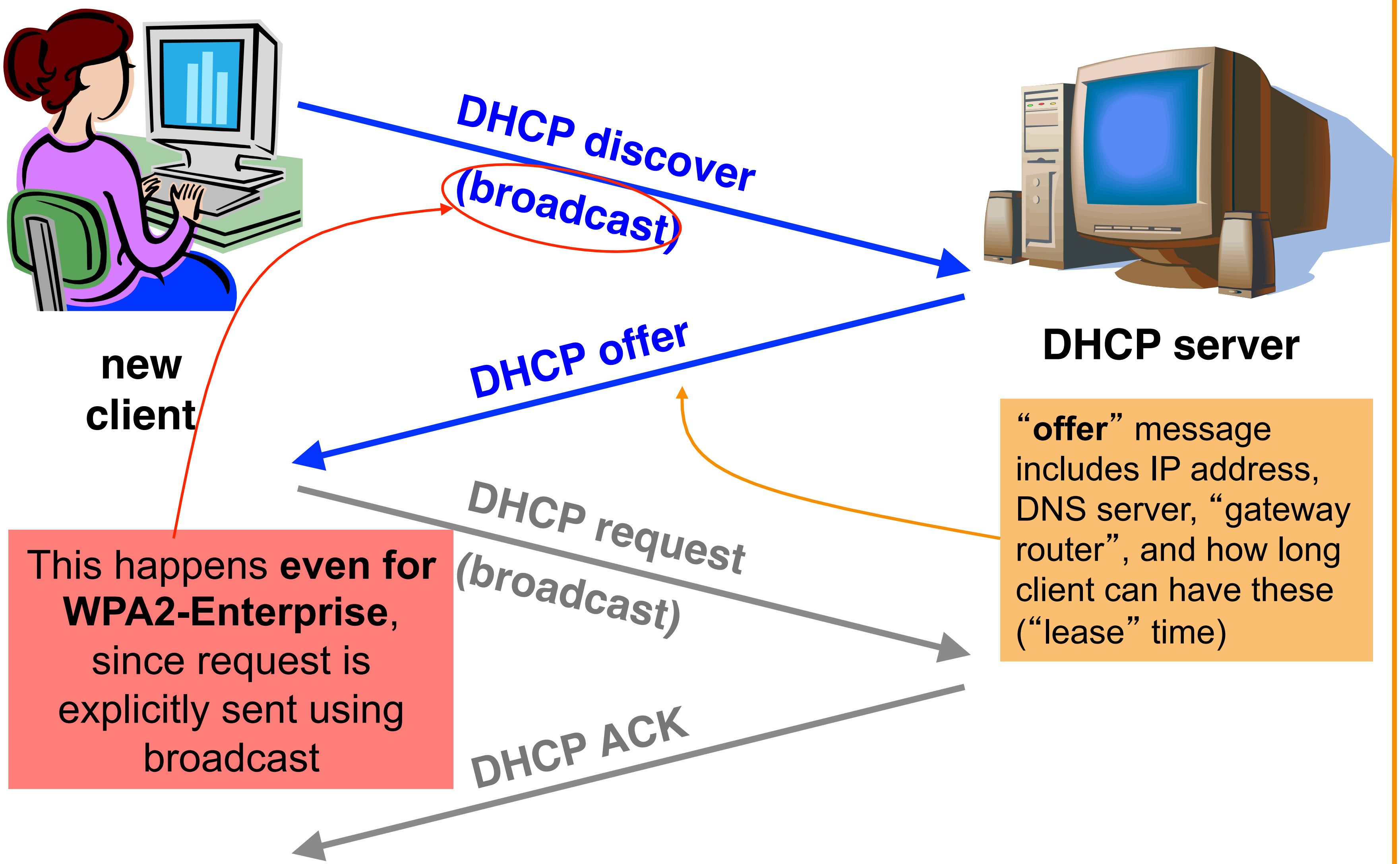
Dynamic Host Configuration Protocol



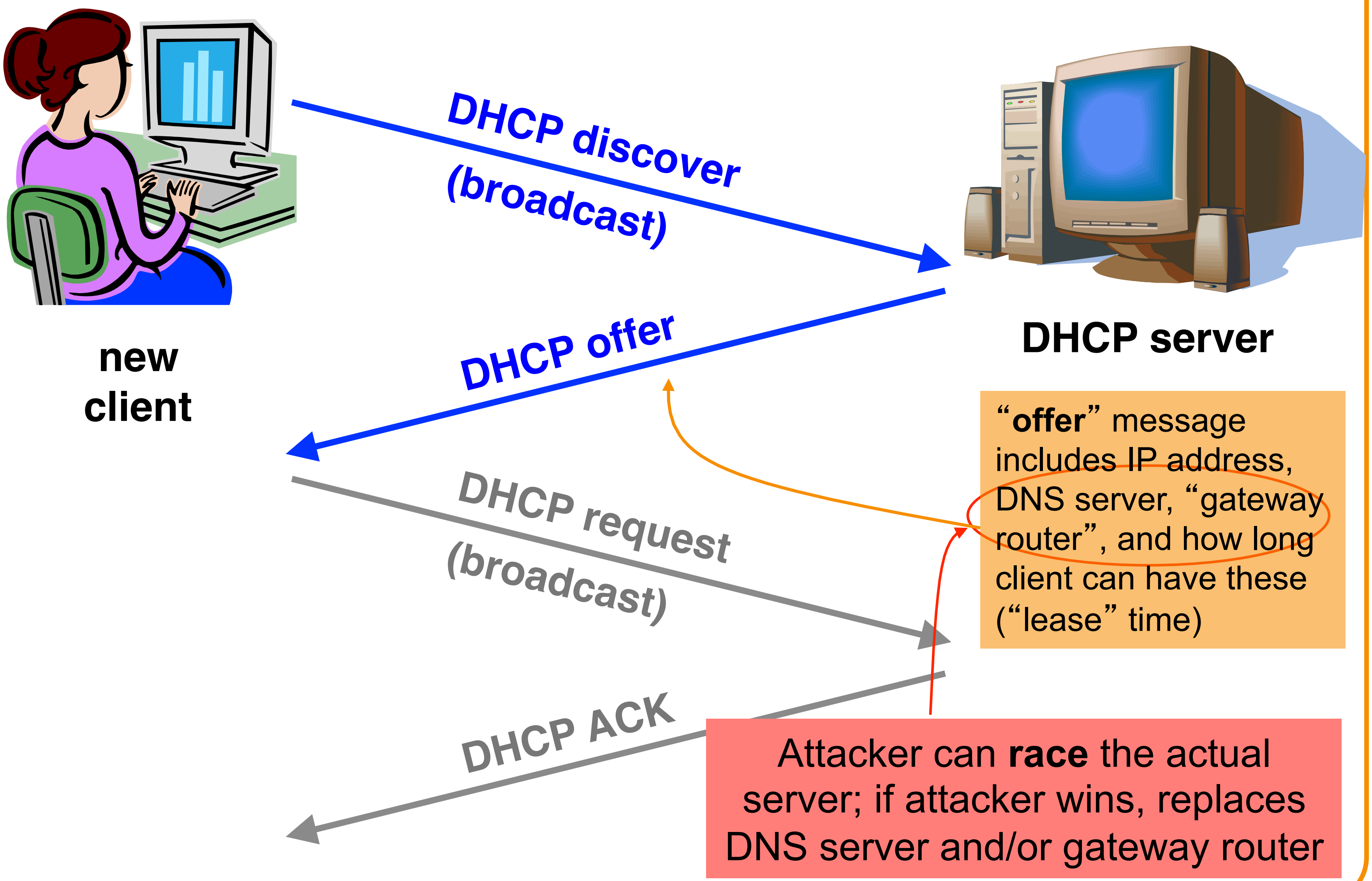
Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol



DHCP Threats

- Substitute a fake DNS server
  - Redirect any of a host's lookups to a machine of attacker's choice (e.g., `gmail.com` = **`6.6.6.6`**)
- Substitute a fake gateway router
  - Intercept all of a host's off-subnet traffic
    - (even if not preceded by a DNS lookup)
  - Relay contents back and forth between host and remote server
    - Modify however attacker chooses
  - This is one type of invisible ***Man In The Middle*** (MITM)
    - Victim host generally has no way of knowing it's happening! 😟
    - (Can't necessarily alarm on peculiarity of receiving multiple DHCP replies, since that can happen benignly)
- How can we fix this?

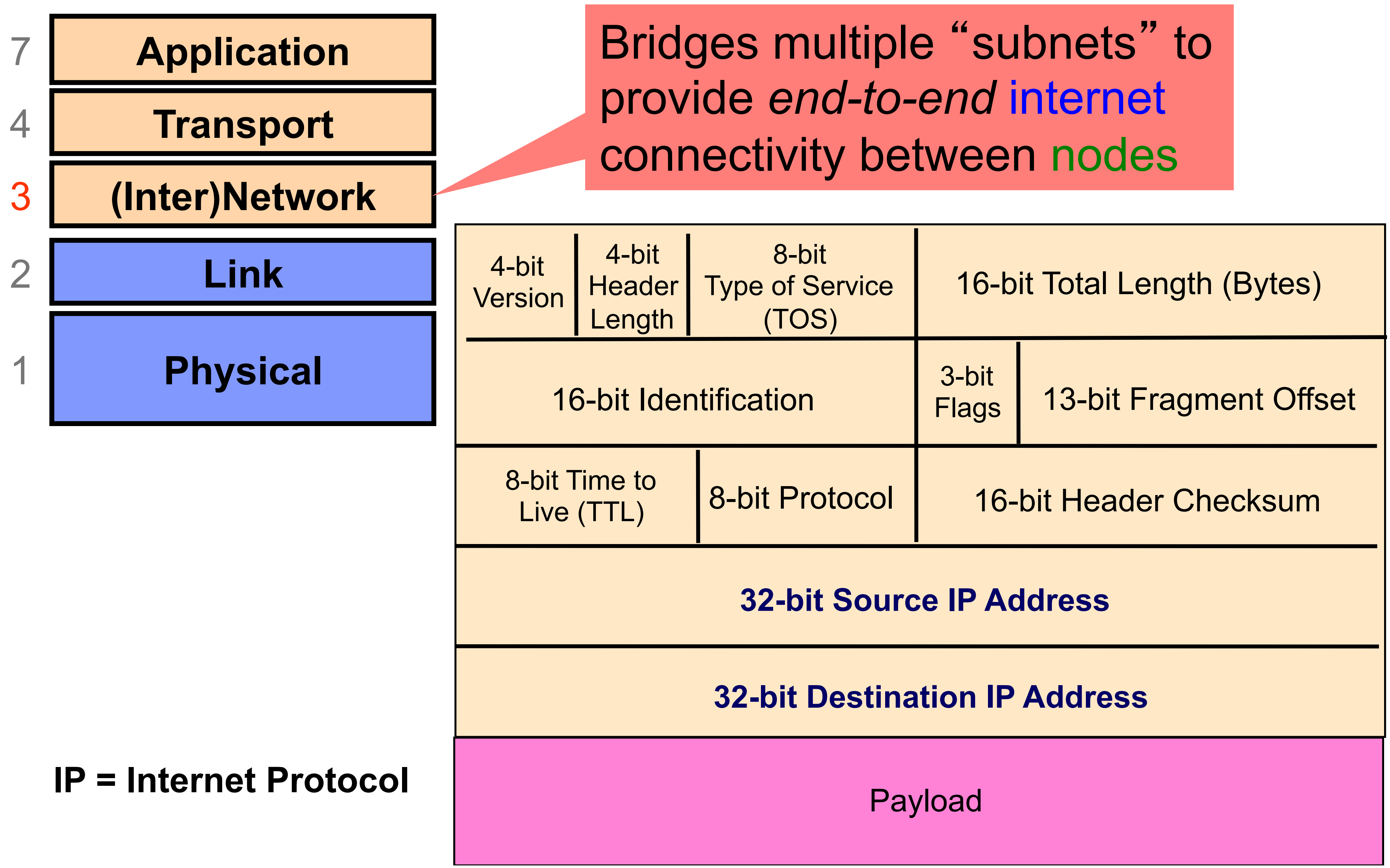
Hard, because we lack a ***trust anchor***

Summary: DHCP Security Issues

- DHCP threats highlight:
  - Broadcast protocols inherently at risk of **local** attacker spoofing
    - Attacker knows exactly when to try it …
    - … and can see the victim's messages
  - When initializing, systems are particularly vulnerable because they can *lack a trusted foundation* to build upon
  - Tension between wiring in trust vs. flexibility and convenience
  - MITM attacks **insidious** because no indicators they're occurring

Layer 3 Threats

Layer 3's View of the World



Network-Layer (IP) Threats

- Can set arbitrary IP source address
  - "Spoofing" - receiver has no idea who attacker is
  - Could be blind, or could be coupled w/ sniffing
  - Note: many attacks require two-way communication
    - So successful off-path/blind spoofing might not suffice
- Can set arbitrary destination address
  - Enables "scanning" - brute force searching for hosts
- Can *send like crazy* (flooding)
  - IP has no general mechanism for tracking overuse
  - IP has no general mechanism for tracking consent
  - Very hard to tell where a spoofed flood comes from!
- **If** attacker can manipulate routing, can bring traffic to them for *eavesdropping* or MITM (not so easy)