

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

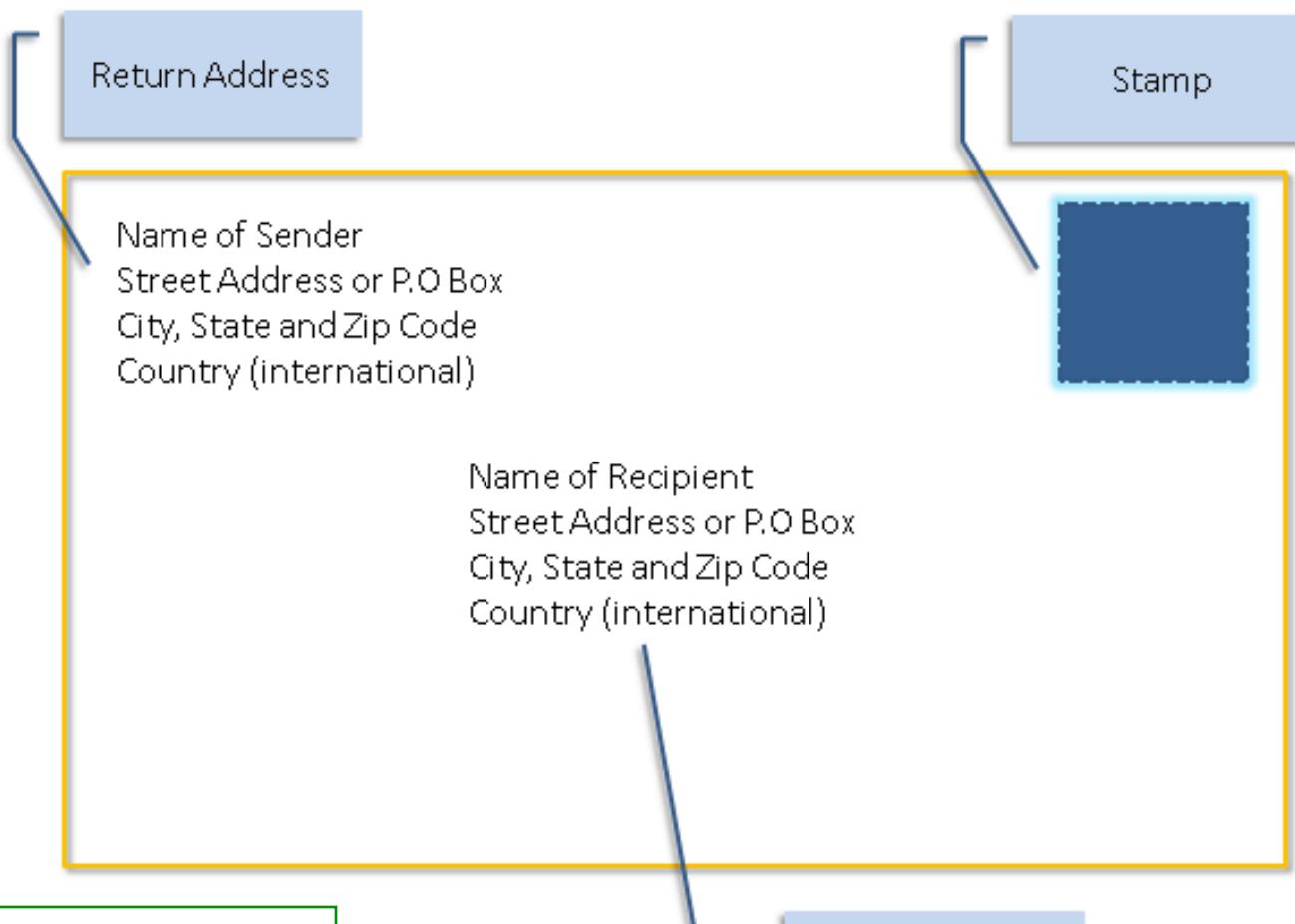
<http://inst.eecs.berkeley.edu/~cs161/>

March 9, 2017

IP Packet Header (Continued)

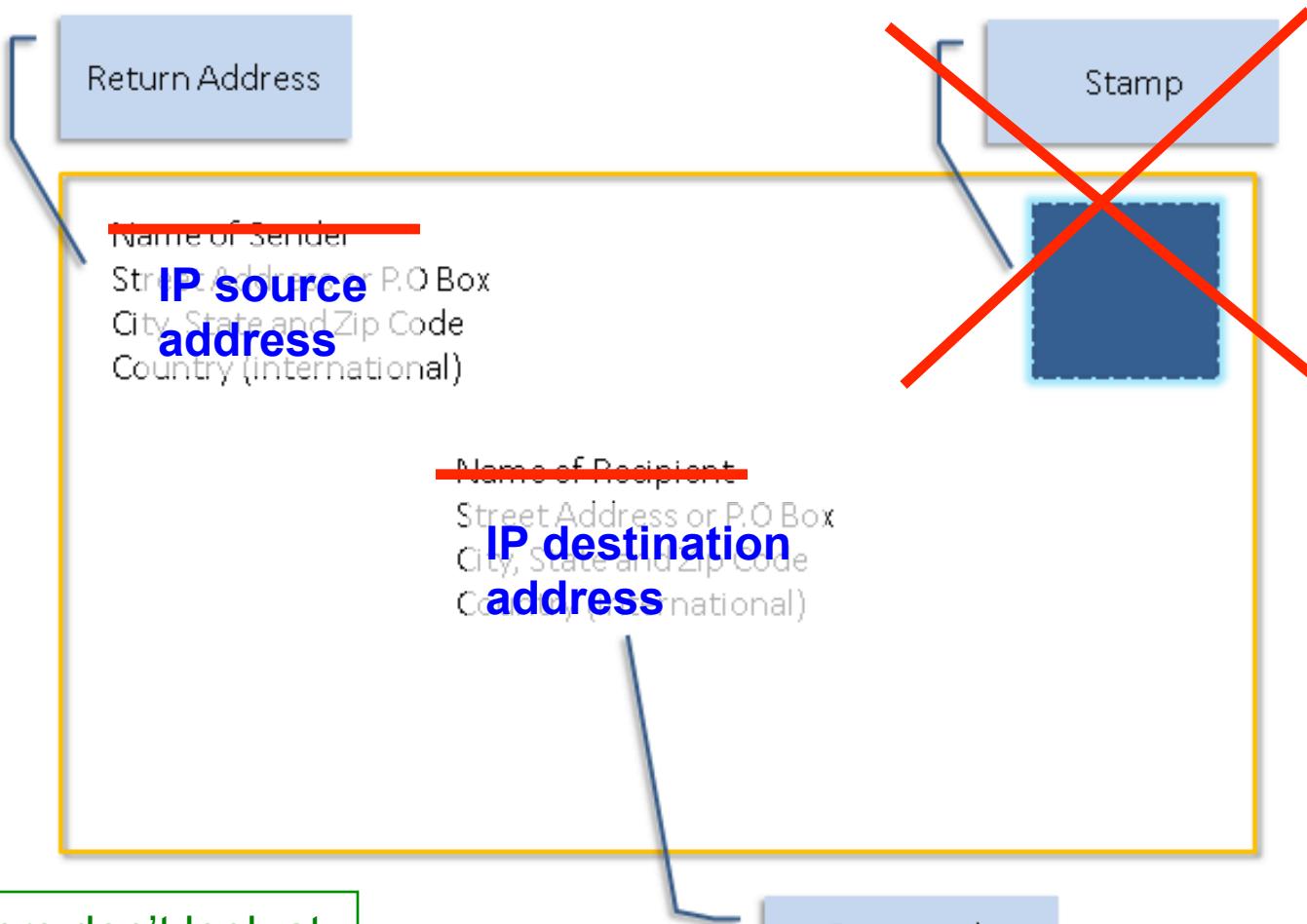
- Two IP addresses
 - Source IP address (32 bits in main IP version)
 - Destination IP address (32 bits, likewise)
 - Destination address
 - Unique identifier/locator for the receiving host
 - Allows each node to make forwarding decisions
 - Source address
 - Unique identifier/locator for the sending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

Postal Envelopes:



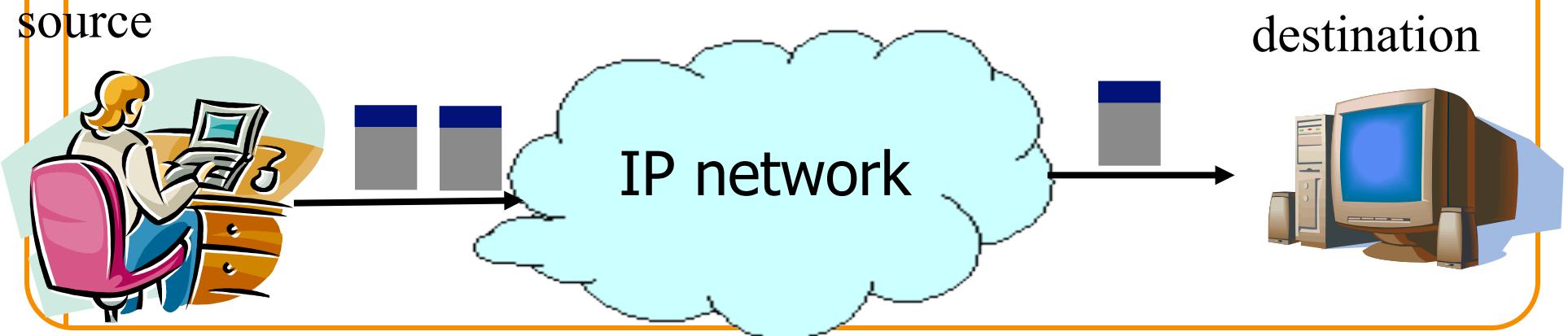
(Post office doesn't
look at the letter
inside the envelope)

Analogy of IP to Postal Envelopes:



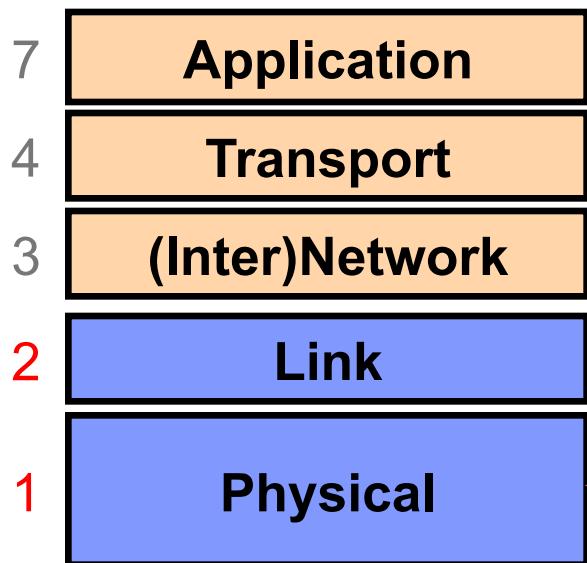
IP: “*Best Effort*” Packet Delivery

- 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?



Framing and transmission of a collection of bits into individual **messages** sent across a single “subnetwork” (one physical technology)

Encoding **bits** to send them over a single physical link
e.g. patterns of
voltage levels / photon intensities / RF modulation

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
 - o 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.

The screenshot shows the Wireshark interface with the following details:

- Title Bar:** all.trace2 [Wireshark 1.6.2]
- Menu Bar:** File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
- Toolbar:** Includes icons for file operations (Open, Save, Print, Copy), search, and various analysis tools.
- Filter Bar:** Contains a "Filter:" input field, an "Expression..." button, a "Clear" button, and an "Apply" button.
- Table View:** A detailed table of captured packets. The columns are: No., Time, Source, Destination, Protocol, Length, and Info. The table lists 13 packets, mostly TCP and HTTP traffic between 10.0.1.9 and 31.13.75.23.
- Packet Details:** A scrollable pane showing the raw hex and ASCII representation of selected packet 10. The hex dump shows the HTTP response code 200 OK.
- Status Bar:** Displays the file path "/Users/vern/tmp/all.trace2", the number of packets (13), the number of displayed packets (13), the number of marked packets (0), the load time (0:00.109), and the profile (Default).

Wireshark: GUI for Packet Capture/Exam.

Wireshark 1.6.2 - all.trace2

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.9	10.0.1.255	BJNP	58	Printer Command: Unknown code (2)
2	0.000198	10.0.1.9	224.0.0.1	BJNP	58	Printer Command: Unknown code (2)
3	2.150663	10.0.1.9	255.255.255.255	DB-LSP-D	172	Dropbox LAN sync Discovery Protocol
4	2.150938	10.0.1.9	10.0.1.255	DB-LSP-D	172	Dropbox LAN sync Discovery Protocol
5	4.514403	10.0.1.13	31.13.75.23	TCP	78	61901 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=4290
6	4.536771	31.13.75.23	10.0.1.13	TCP	74	http > 61901 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK
7	4.536896	10.0.1.13	31.13.75.23	TCP	66	61901 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=429017456 T
8	4.537429	10.0.1.13	31.13.75.23	HTTP	590	GET / HTTP/1.1
9	4.553947	31.13.75.23	10.0.1.13	TCP	66	http > 61901 [ACK] Seq=1 Ack=525 Win=15872 Len=0 TSval=1765827012
10	4.626447	31.13.75.23	10.0.1.13	HTTP	600	HTTP/1.1 302 Found
11	4.626579	10.0.1.13	31.13.75.23	TCP	66	61901 > http [ACK] Seq=525 Ack=535 Win=524280 Len=0 TSval=4290174
12	7.065664	10.0.1.9	10.0.1.255	BJNP	58	Printer Command: Unknown code (2)
13	7.065846	10.0.1.9	224.0.0.1	BJNP	58	Printer Command: Unknown code (2)

Frame 10: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits)
Ethernet II, Src: Apple_fe:aa:41 (00:25:00:fe:aa:41), Dst: Apple_41:eb:00 (e4:ce:8f:41:eb:00)
Internet Protocol Version 4, Src: 31.13.75.23 (31.13.75.23), Dst: 10.0.1.13 (10.0.1.13)
Transmission Control Protocol, Src Port: http (80), Dst Port: 61901 (61901), Seq: 1, Ack: 525, Len: 534

Source port: http (80)
Destination port: 61901 (61901)
[Stream index: 0]
Sequence number: 1 (relative sequence number)
[Next sequence number: 535 (relative sequence number)]
Acknowledgement number: 525 (relative ack number)
Header length: 32 bytes
Flags: 0x18 (PSH, ACK)
Window size value: 31
[Calculated window size: 15872]
[Window size scaling factor: 512]
Checksum: 0xf42f [validation disabled]

Hex	Dec	ASCII
0000	e4 ce 8f 41 eb 00 00 25 00 fe aa 41 08 00 45 20	...A...% ...A..E
0010	02 4a 67 be 00 00 58 06 83 9f 1f 0d 4b 17 0a 00	Jg...X.K...
0020	01 0d 00 50 f1 cd d5 b8 c0 31 96 68 cb 28 80 18	...P.... .1.h.(..
0030	00 1f f4 2f 00 00 01 01 08 0a 69 40 62 0b 19 92	.../.... .i@b...
0040	49 70 48 54 54 50 2f 31 2e 31 20 33 30 32 20 46	IpHTTP/1.1 302 F

Frame (frame), 600 bytes Packets: 13 Displayed: 13 Marked: 0 Load time: 0:00.109 Profile: Default

Wireshark: GUI for Packet Capture/Exam.

The screenshot shows the Wireshark interface with the following details:

- Title Bar:** all.trace2 [Wireshark 1.6.2]
- Menu Bar:** File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
- Toolbar:** Includes icons for file operations (New, Open, Save, Print, Export), search, and various analysis tools.
- Filter Bar:** Contains a "Filter:" input field, an "Expression..." button, a "Clear" button, and an "Apply" button.
- Table View:** Displays a list of network packets. The columns are No., Time, Source, Destination, Protocol, Length, and Info. The table shows 13 entries, mostly from source 10.0.1.9 to destination 10.0.1.255, with some traffic to 224.0.0.1 and 31.13.75.23. The last two entries are from 10.0.1.9 to 224.0.0.1.
- Details View:** Shows the details of Frame 10, which is an HTTP/1.1 302 Found response. The response includes headers such as Location, P3P, Set-Cookie, Content-Type, X-FB-Debug, Date, Connection, and Content-Length.
- Hex View:** Shows the raw hex dump of the selected frame, starting with 0000 and ending with 0040.
- Statistics View:** Shows the number of frames (13) and the load time (0:00.109).
- Profile View:** Shows the profile as Default.

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.

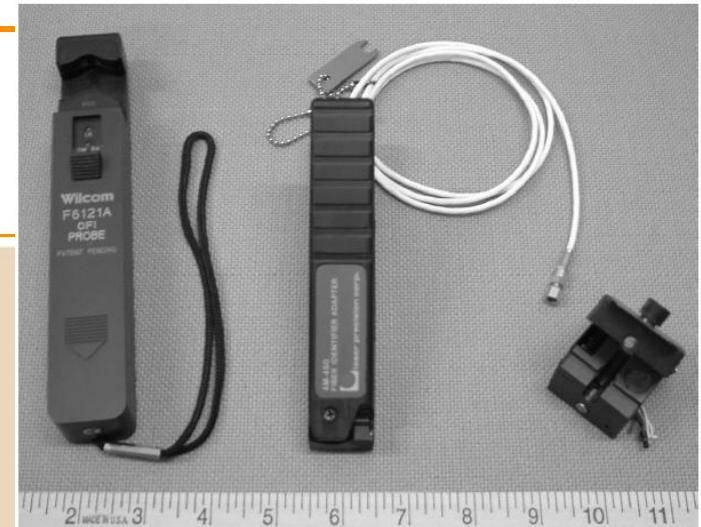
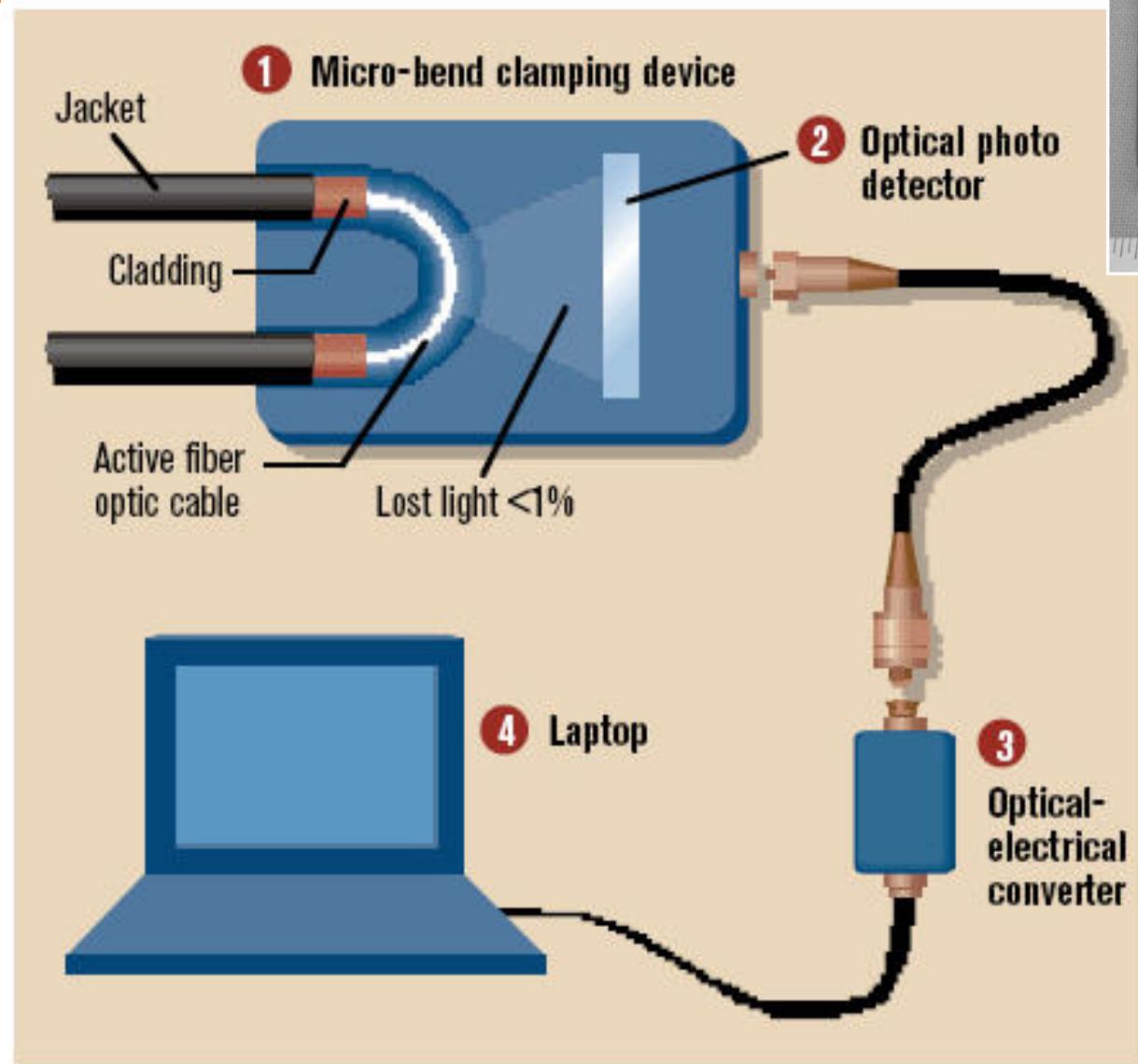
Upon their return to the United States, intelligence agents from the NSA analyzed the recordings and tried to decipher any encrypted information. The Soviets apparently were confident in the security of their communications lines, as a surprising amount of sensitive information traveled through the lines 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!

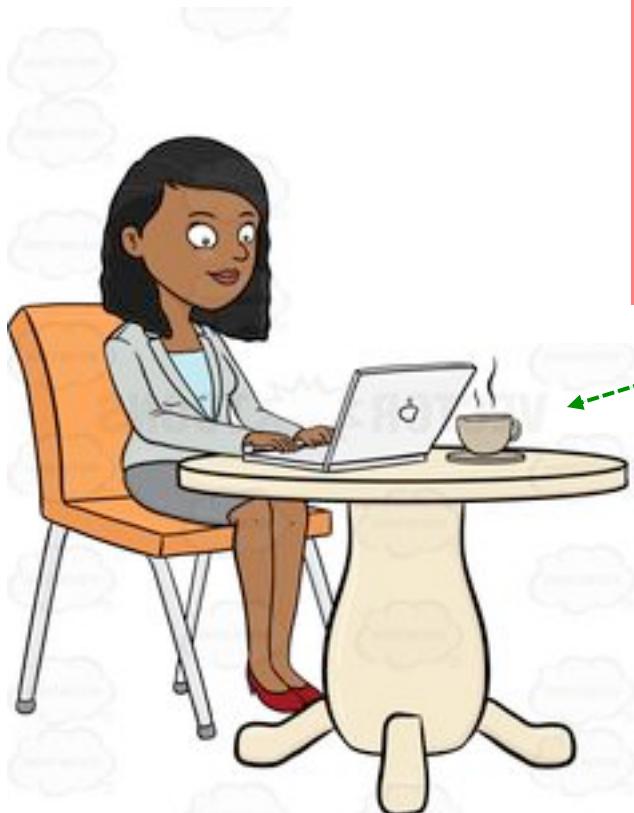
Stealing Photons



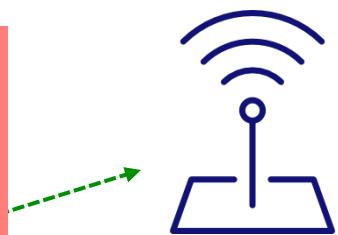
Protecting Against Eavesdropping in the Coffee Shop

Coffee Shop

1. Join the wireless network

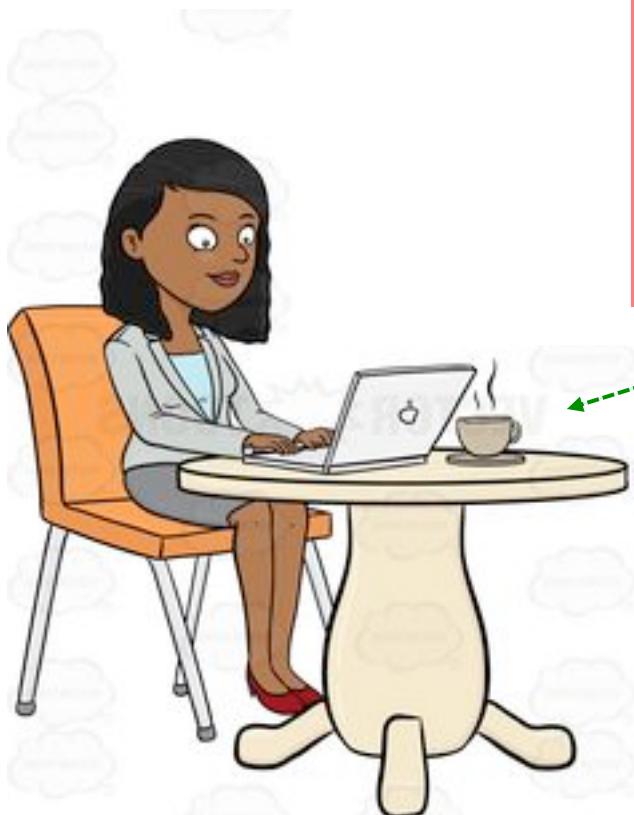


If either match up, your laptop joins the network.
Optionally performs a cryptographic exchange.

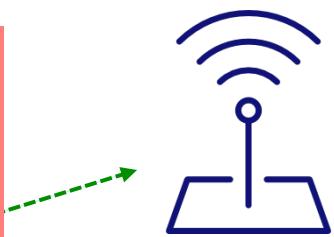


Coffee Shop

1. Join the wireless network

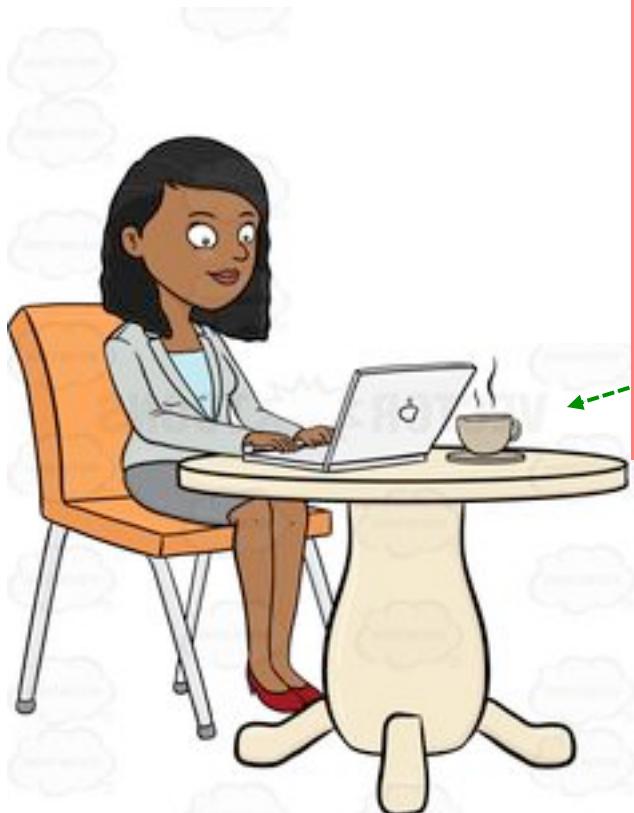


If either match up, your laptop joins the network.
Optionally performs a cryptographic exchange.

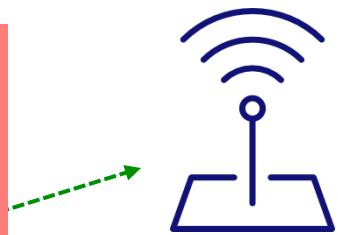


Coffee Shop

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



KeyCounter
(and other stuff)

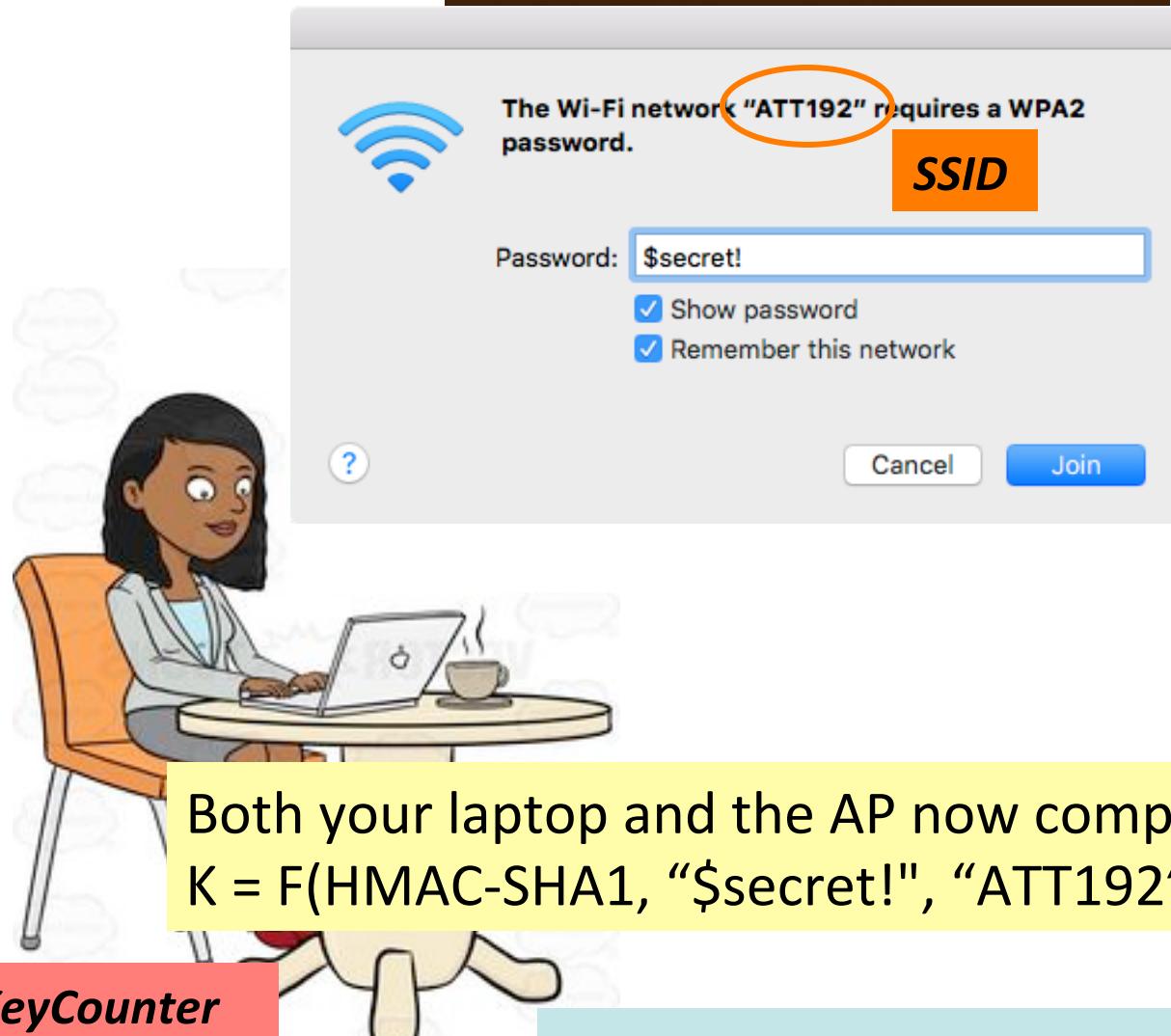


Password: \$secret!

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

Coffee Shop



The Wi-Fi network "ATT192" requires a WPA2 password.

SSID

Password: \$secret!

- Show password
- Remember this network

Cancel

Join

Both your laptop and the AP now compute:
 $K = F(\text{HMAC-SHA1}, "\$secret!", "ATT192", \text{KeyCounter}, 4096)$

KeyCounter
(and other stuff)

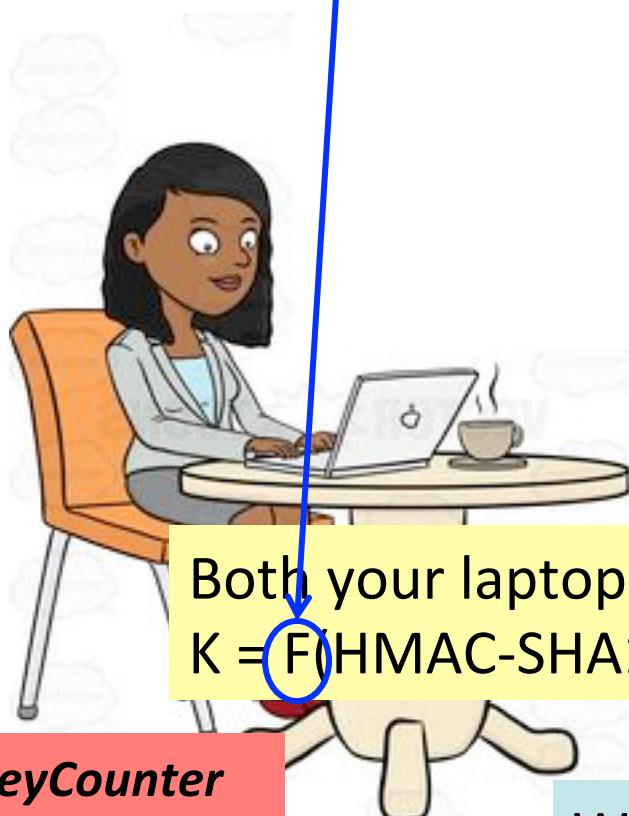
WPA2, common form ("Personal"; simplified)



Password: \$secret!

KeyCounter
(and other stuff)

Coffee Shop



This function



Password: \$secret!

KeyCounter
(and other stuff)

Both your laptop and the AP now compute:

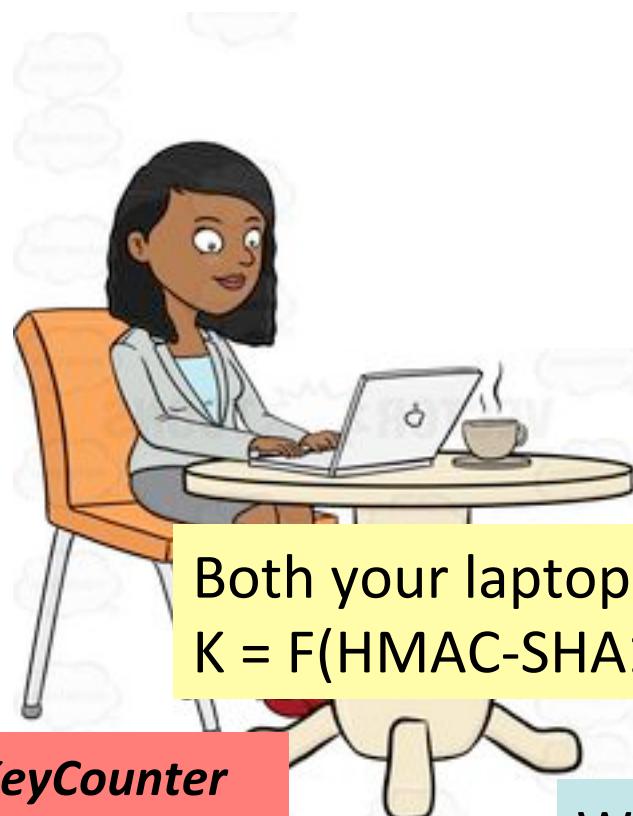
$K = F(\text{HMAC-SHA1}, "\$secret!", \text{"ATT192"}, \text{KeyCounter}, 4096)$

KeyCounter
(and other stuff)

WPA2, common form (“Personal”; simplified)

Coffee Shop

This function
computes **this** many iterations



Password: \$secret!

KeyCounter
(and other stuff)

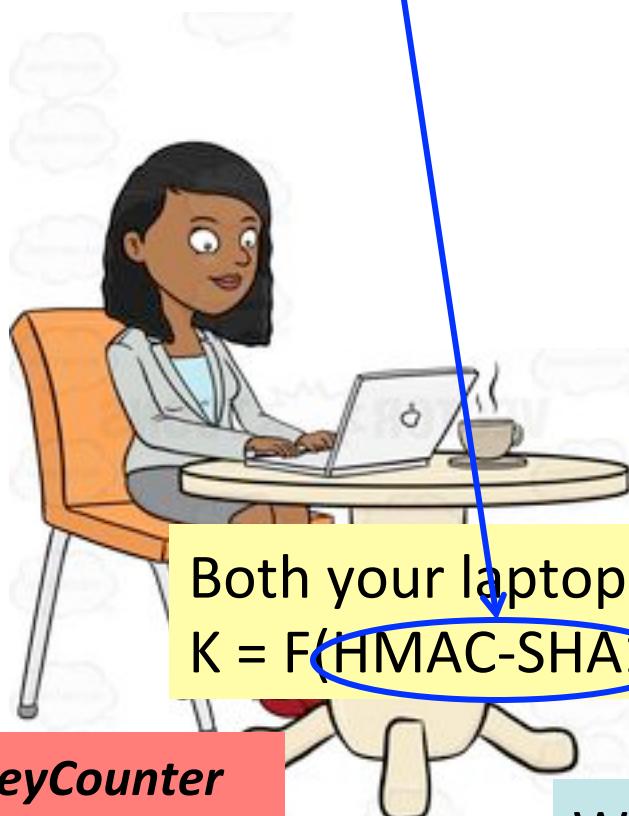
Both your laptop and the AP now compute:
 $K = F(\text{HMAC-SHA1}, "\$secret!", "ATT192", \text{KeyCounter}, 4096)$

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:

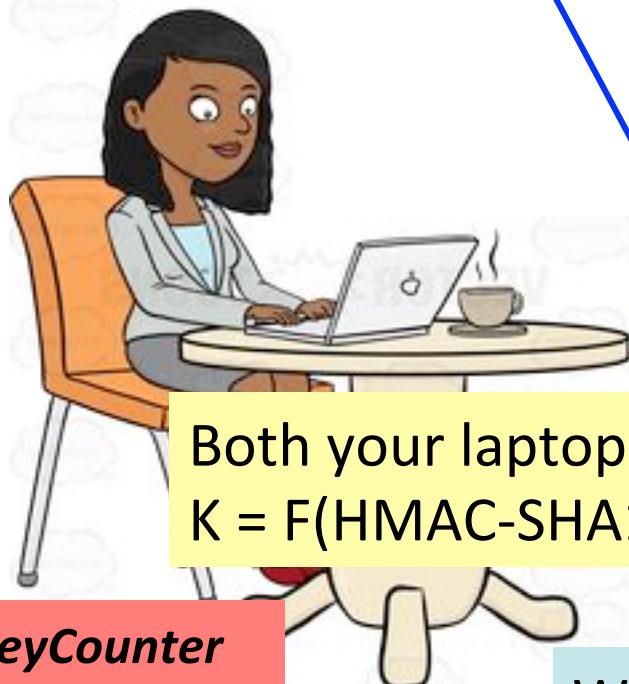
$K = F(\text{HMAC-SHA1}, "\$secret!", \text{"ATT192"}, \text{KeyCounter}, 4096)$

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)

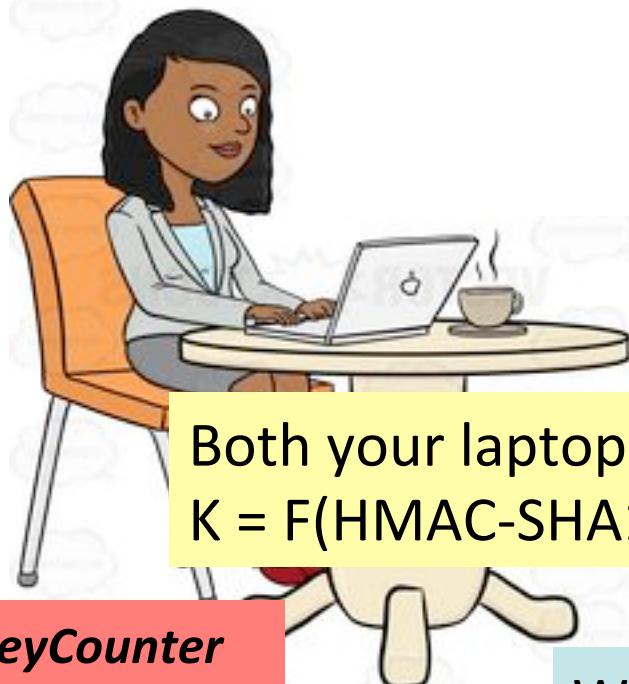
Both your laptop and the AP now compute:
 $K = F(\text{HMAC-SHA1}, \text{"$secret!"}, \text{"ATT192"}, \text{KeyCounter}, 4096)$

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:

$K = F(\text{HMAC-SHA1}, "\$secret!", \text{"ATT192"}, \text{KeyCounter}, 4096)$

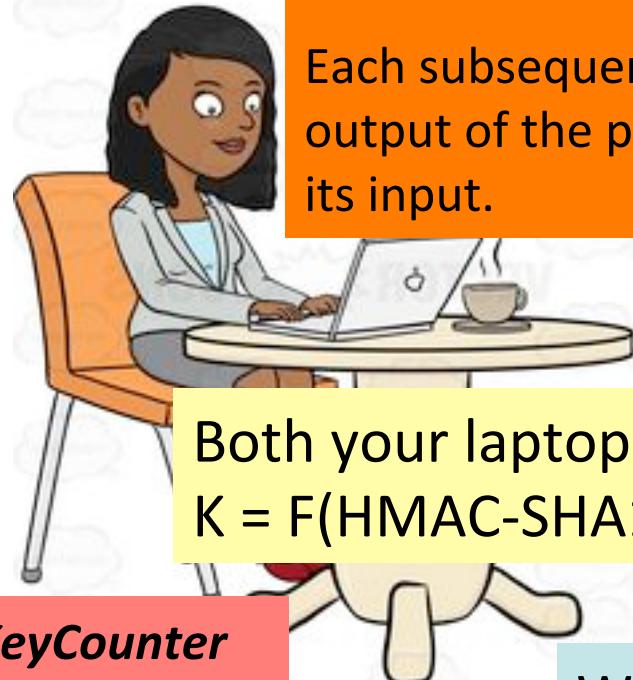
KeyCounter
(and other stuff)

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.



KeyCounter
(and other stuff)

Both your laptop and the AP now compute:
 $K = F(\text{HMAC-SHA1}, "\$secret!", "ATT192", \text{KeyCounter}, 4096)$



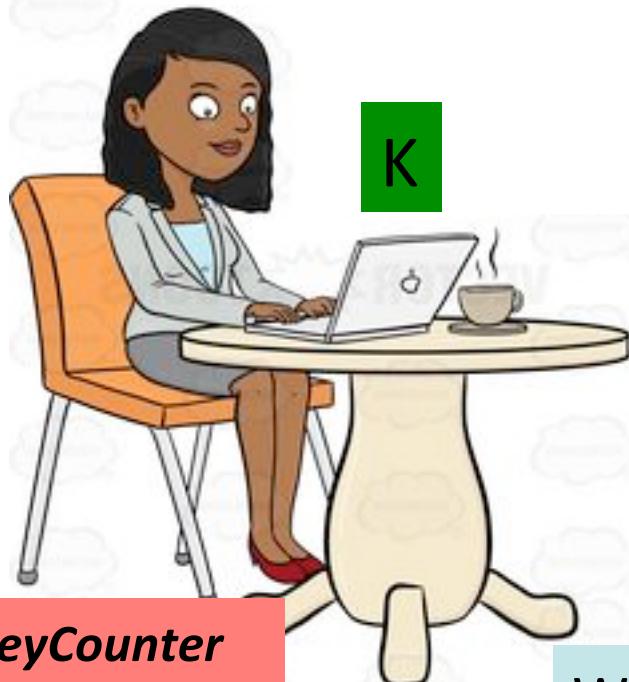
Password: **\$secret!**

KeyCounter
(and other stuff)

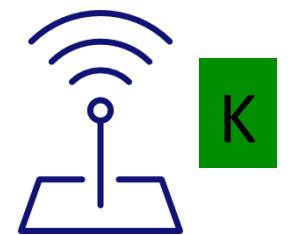
WPA2, common form (“Personal”; simplified)

Coffee Shop

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



KeyCounter
(and other stuff)



Password: \$secret!

KeyCounter
(and other stuff)

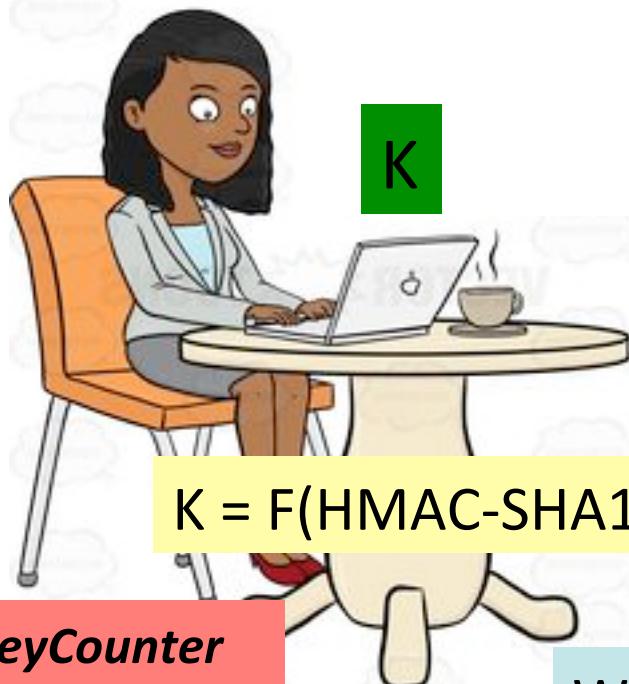
WPA2, common form (“Personal”; simplified)



Eve

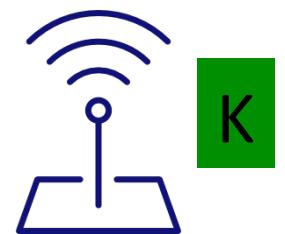


*Eve
attacks!*



KeyCounter
(and other stuff)

$K = F(\text{HMAC-SHA1}, "\$secret!", \text{"ATT192"}, \text{KeyCounter}, 4096)$



Password: \$Secret!

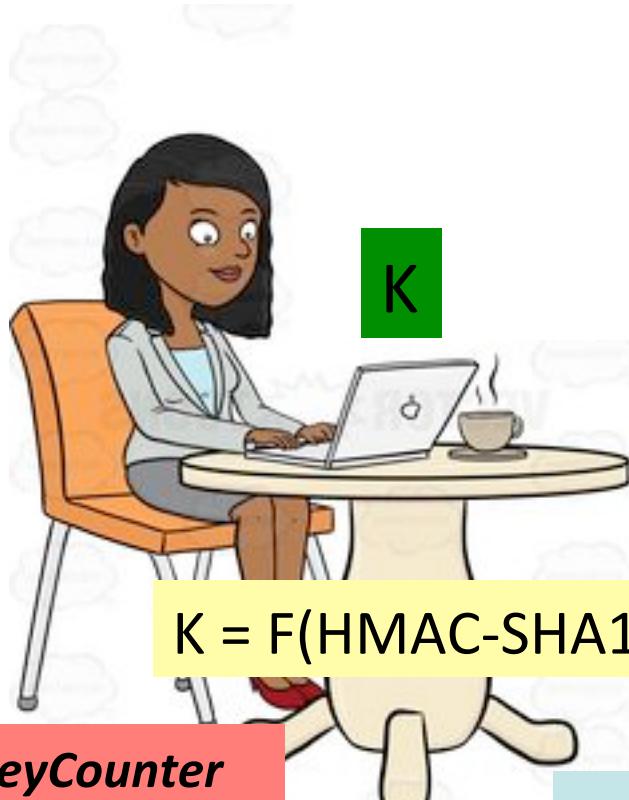
KeyCounter
(and other stuff)

WPA2, common form ("Personal"; simplified)



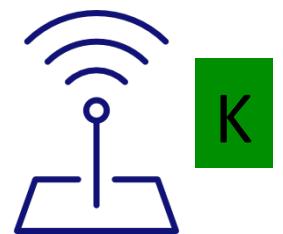
Eve

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.



KeyCounter
(and other stuff)

$K = F(\text{HMAC-SHA1}, "\$secret!", \text{"ATT192"}, \text{KeyCounter}, 4096)$



Password: \$secret!

KeyCounter
(and other stuff)

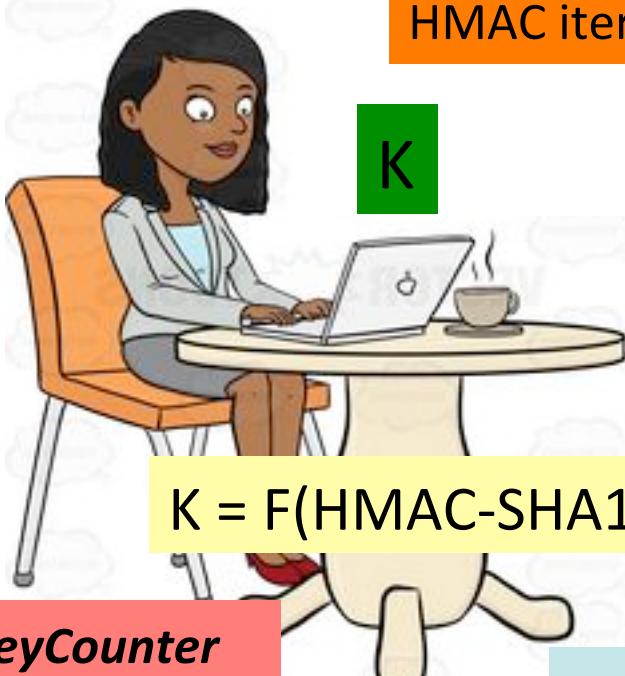
WPA2, common form ("Personal"; simplified)



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.



KeyCounter
(and other stuff)

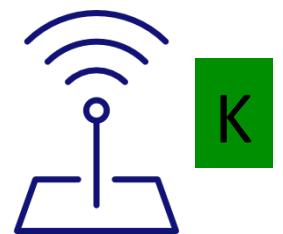


Password: \$secret!

KeyCounter
(and other stuff)

$K = F(\text{HMAC-SHA1}, "\$secret!", "ATT192", \text{KeyCounter}, 4096)$

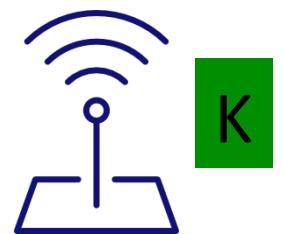
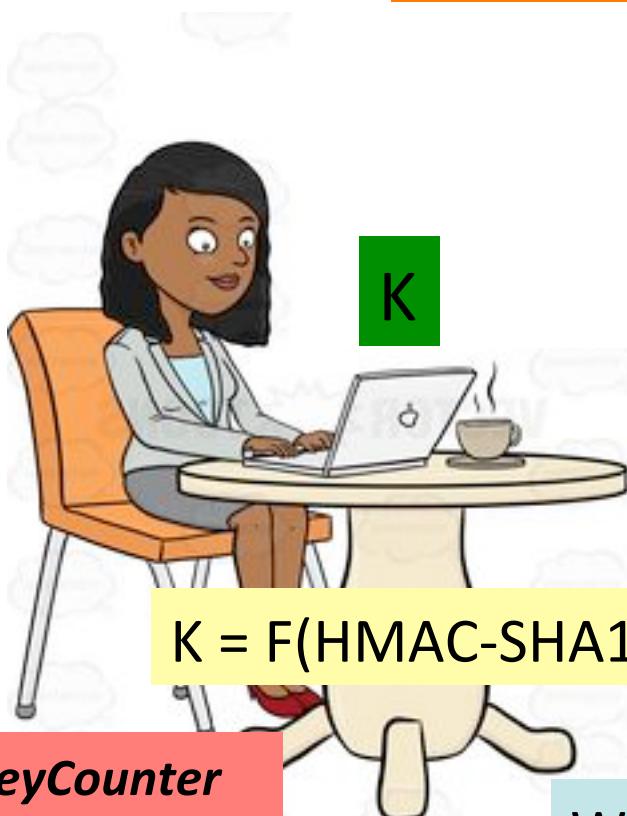
WPA2, common form ("Personal"; simplified)





Eve

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



Password: \$secret!

KeyCounter
(and other stuff)

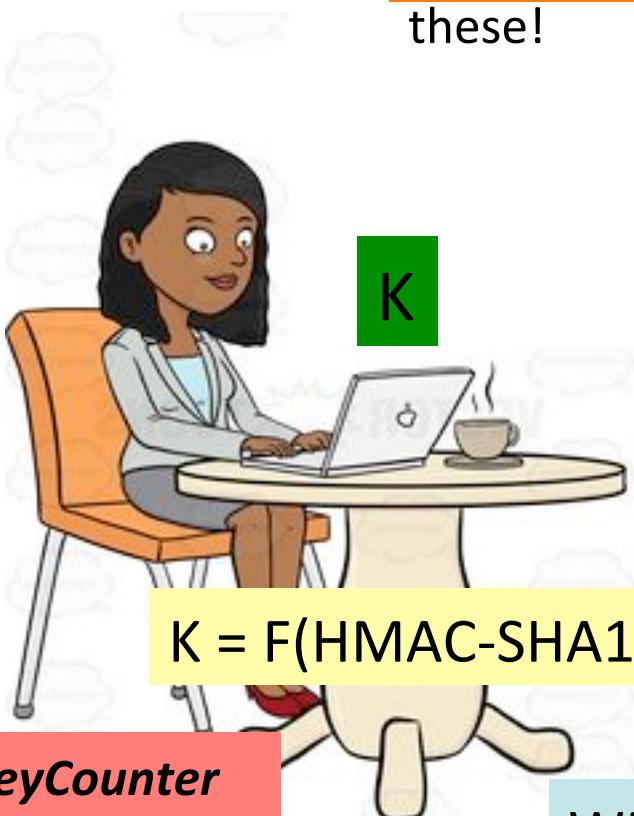
$K = F(\text{HMAC-SHA1}, "\$secret!", \text{"ATT192"}, \text{KeyCounter}, 4096)$

KeyCounter
(and other stuff)

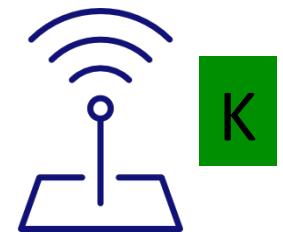
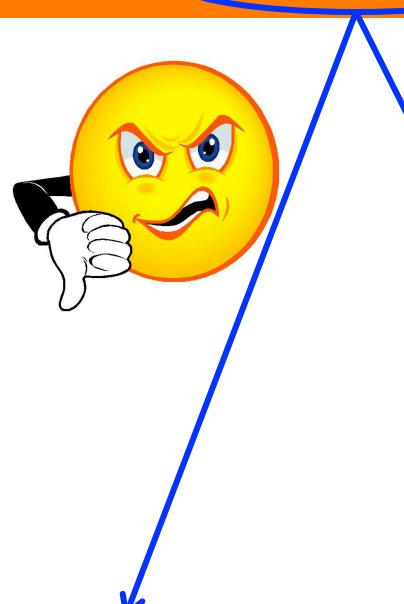
WPA2, common form ("Personal"; simplified)



Eve



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!



Password: \$secret!

KeyCounter
(and other stuff)

$K = F(\text{HMAC-SHA1}, "\$secret!", "ATT192", \text{KeyCounter}, 4096)$

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**.



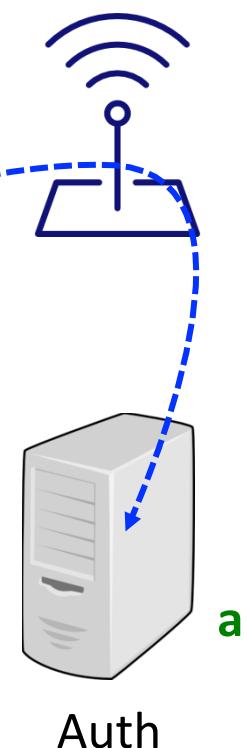
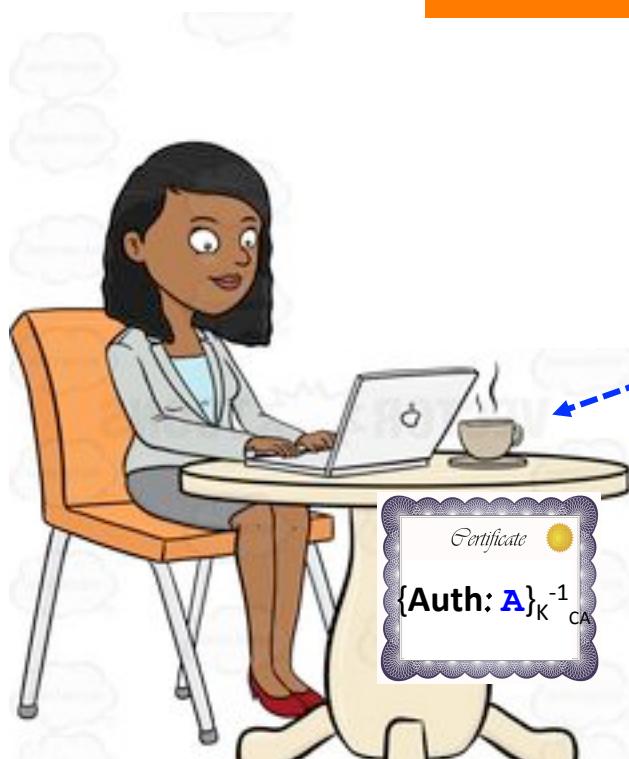
a

Auth

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

Coffee Shop

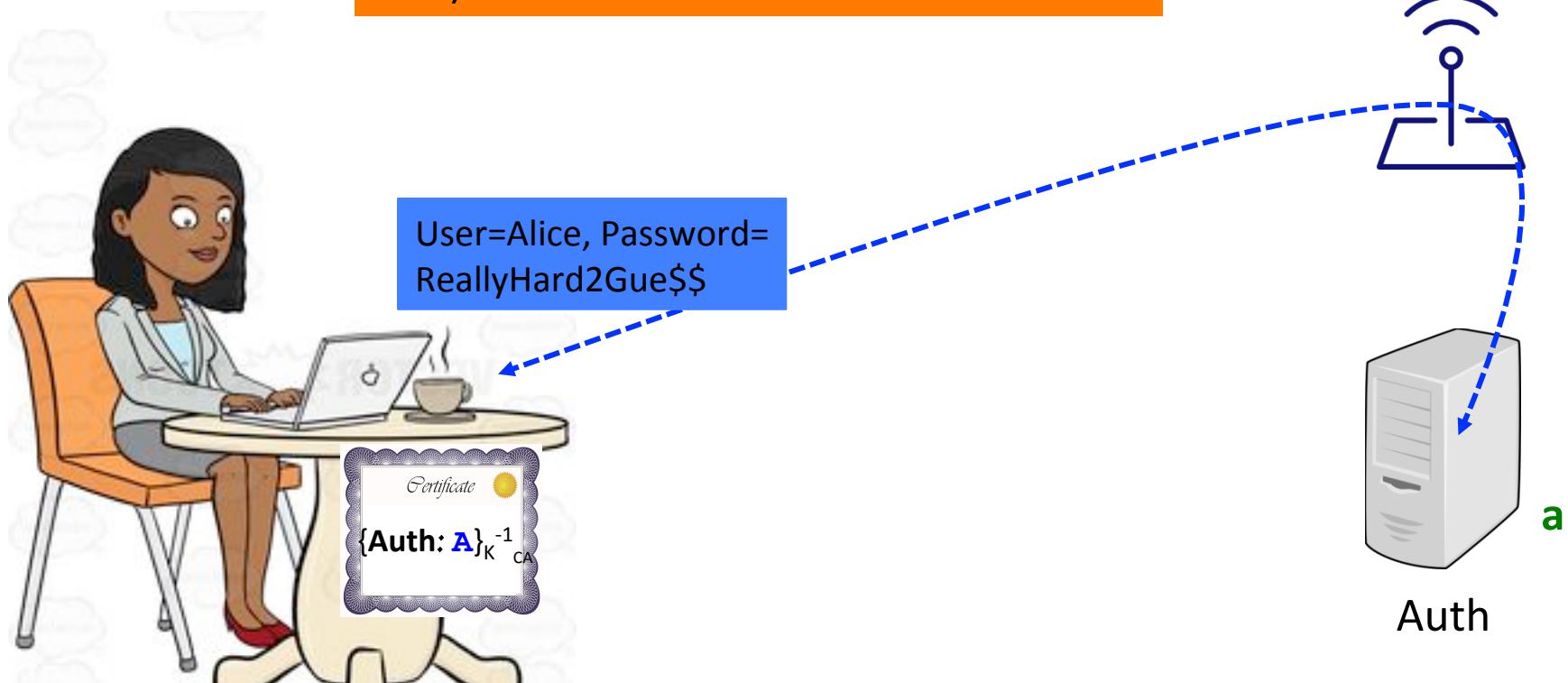
You establish a secure connection via the AP to the Authentication Server using TLS.



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



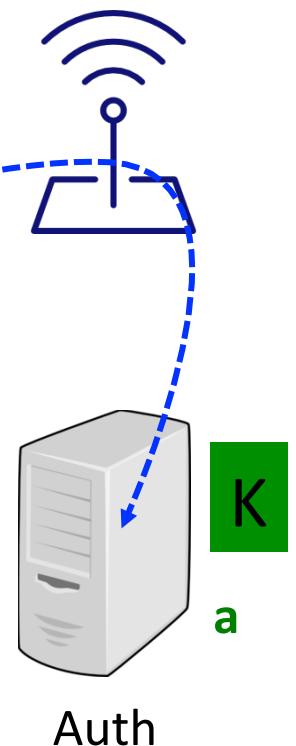
WPA2, actually-secure-but-inconvenient form ("Enterprise"; simplified)

Coffee Shop

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



Certificate
 $\{Auth: A\}_{K_{CA}}^{-1}$



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



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
 - o We care about the **work factor**: how hard is this
 - Sometimes they can just **brute force**
 - o 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”

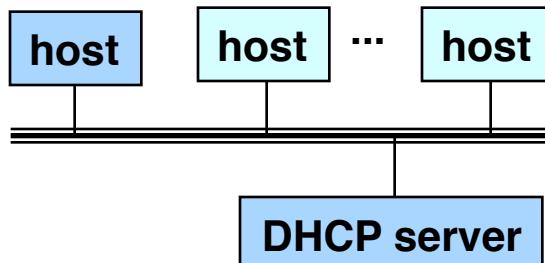


2. Configure your connection



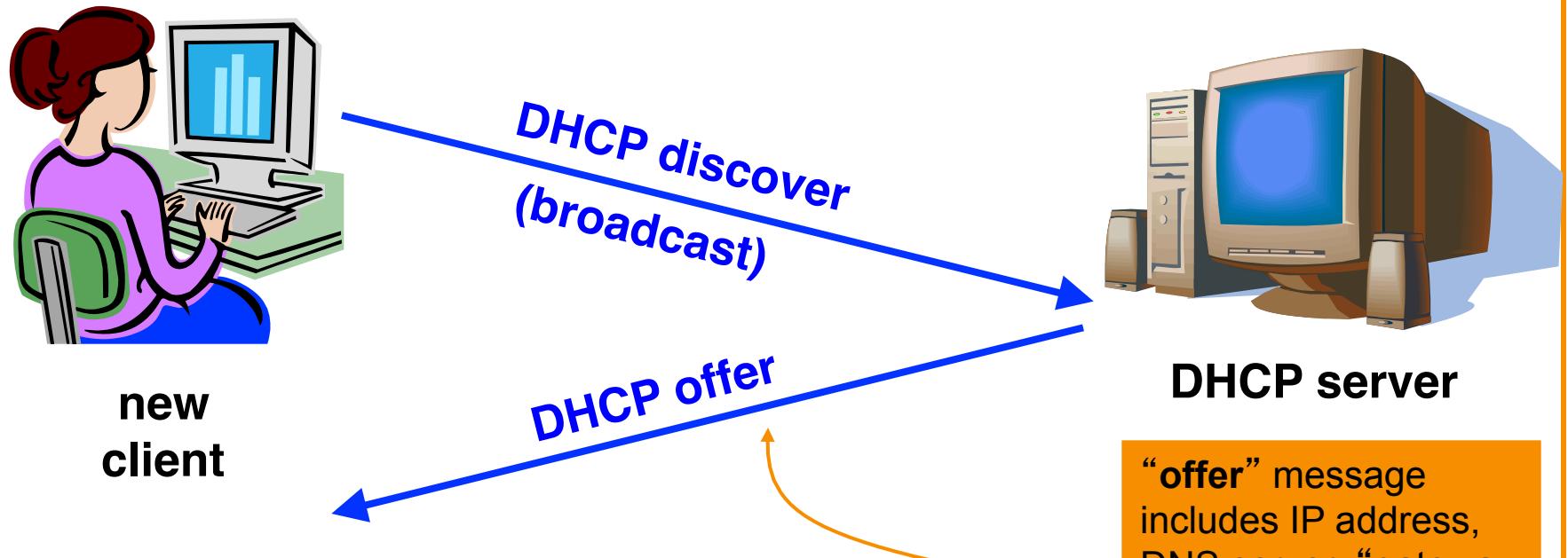
Internet Bootstrapping: DHCP

- 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

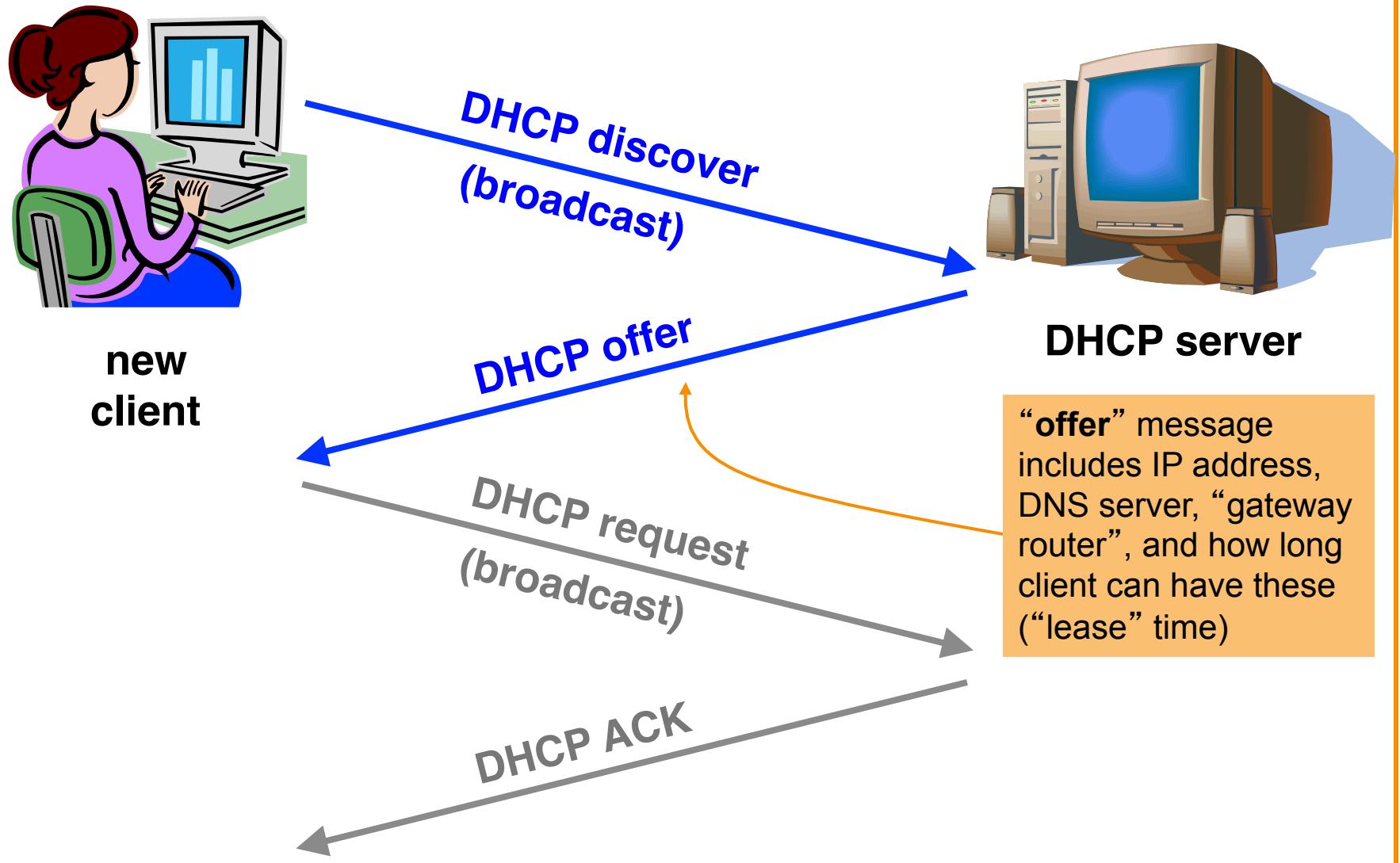
DHCP server

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

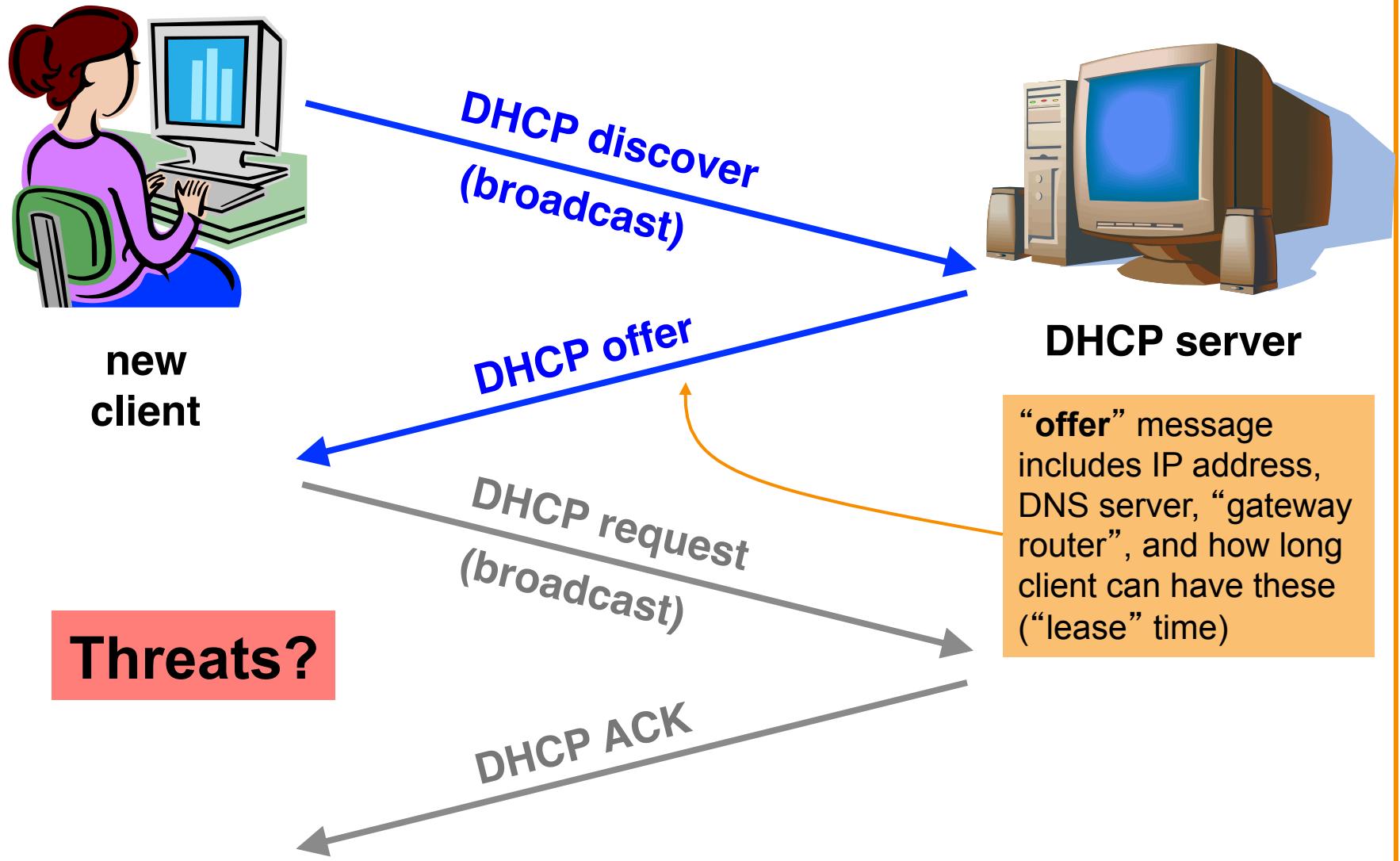
“offer” message includes IP address, DNS server, “gateway router”, and how long client can have these (“lease” time)

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

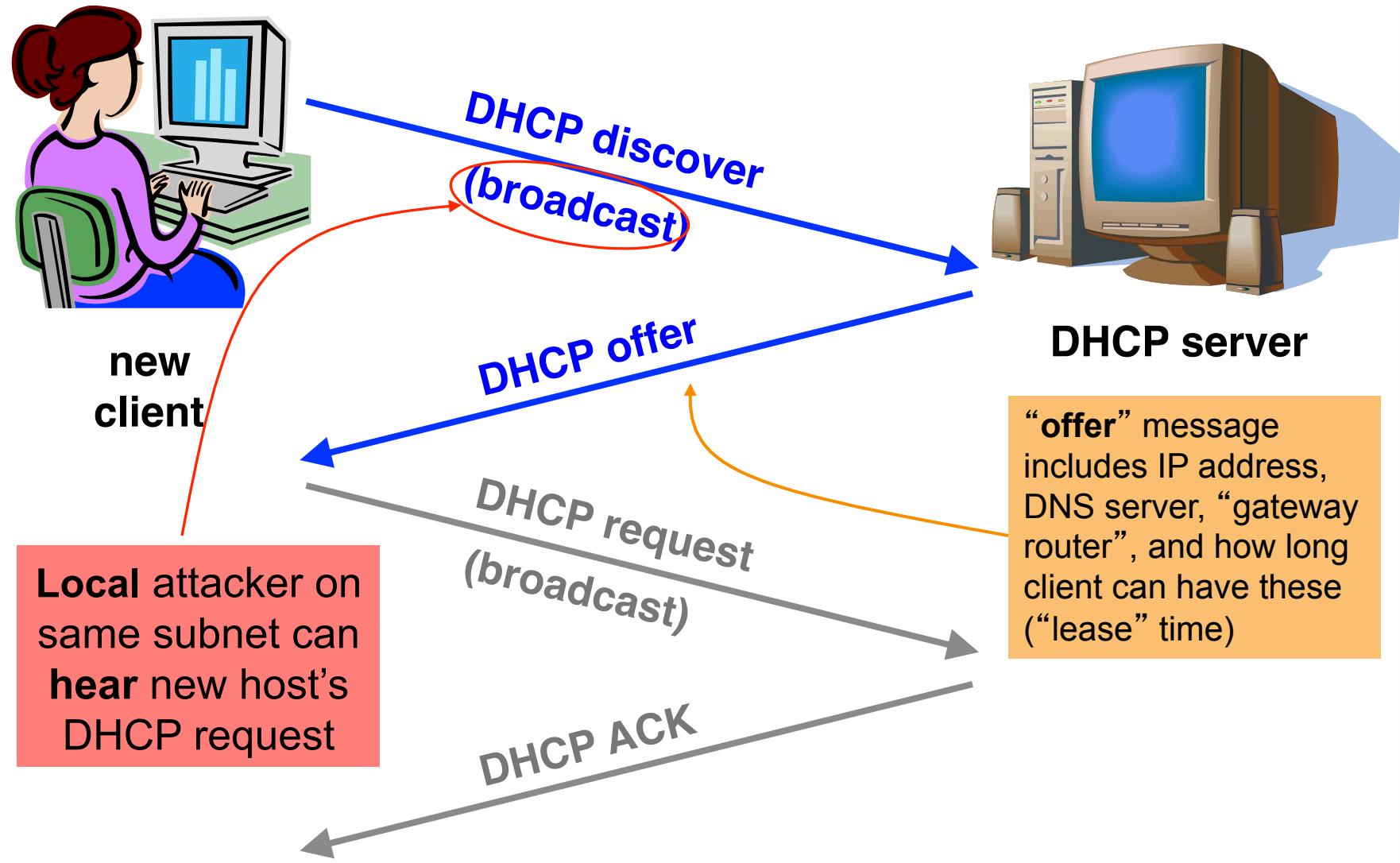
Dynamic Host Configuration Protocol



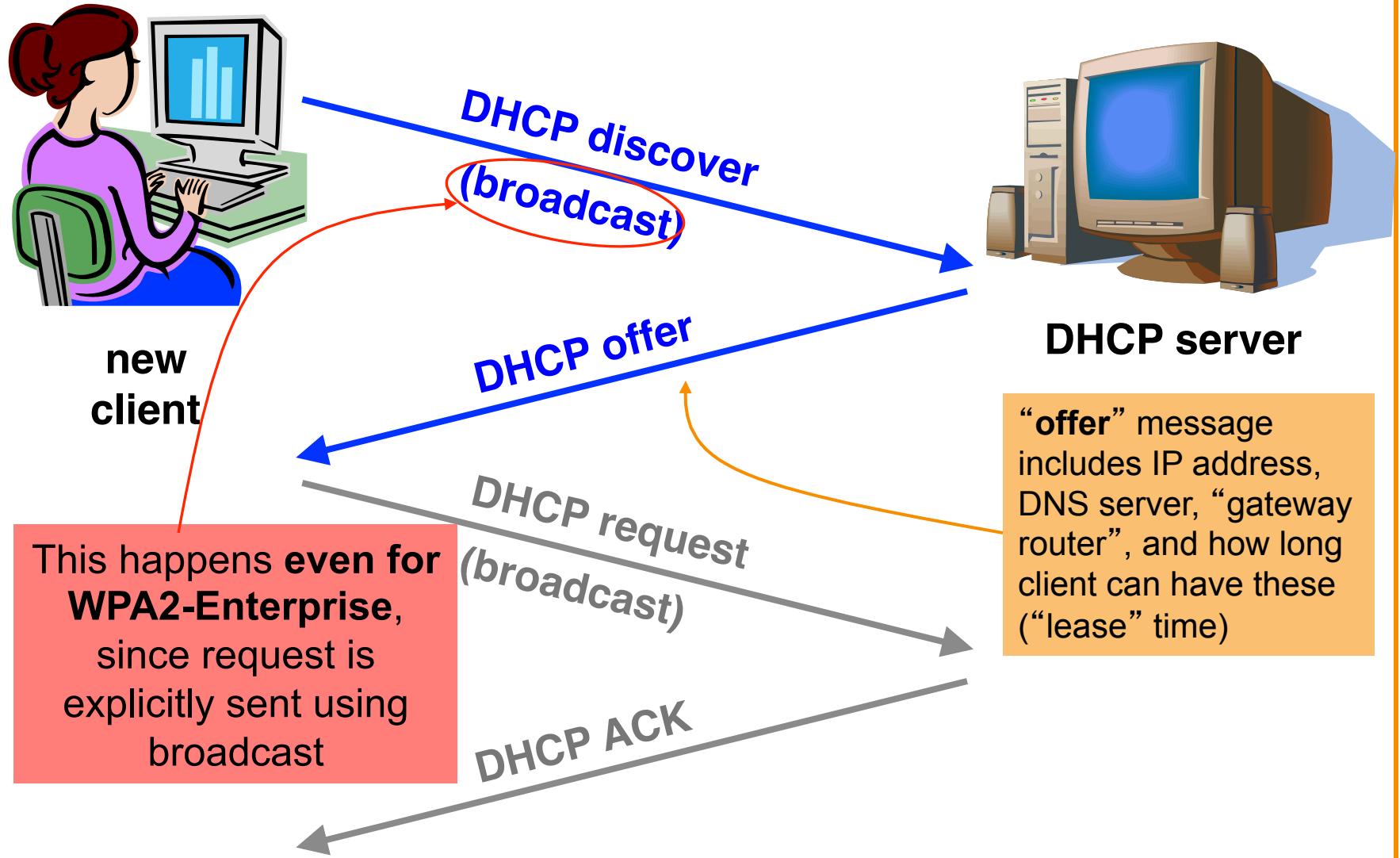
Dynamic Host Configuration Protocol



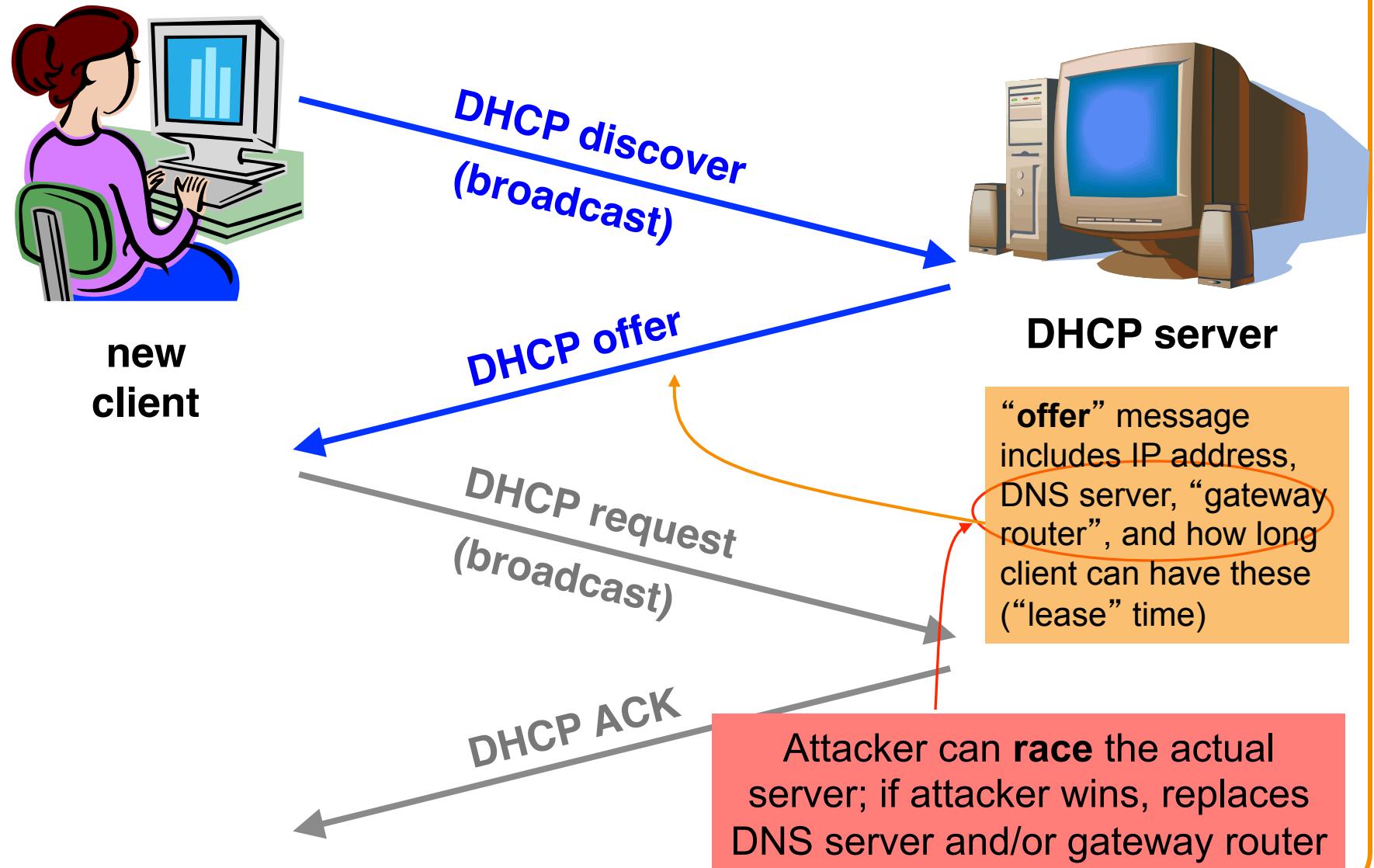
Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol



DHCP Threats

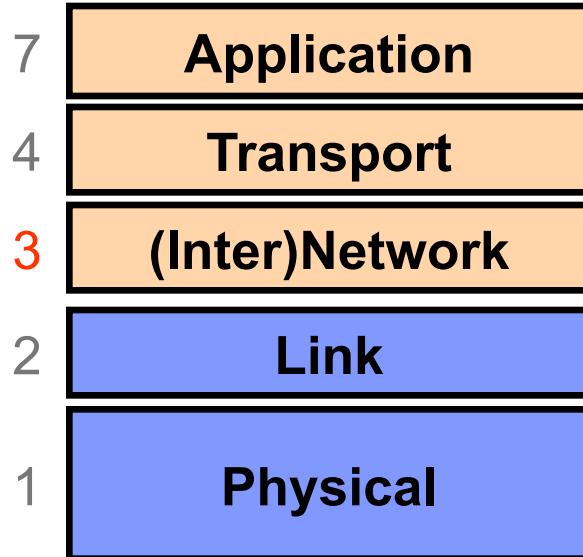
- Substitute a fake DNS server
 - Redirect **any** of a host's lookups to a machine of attacker's choice (e.g., **gmail.com** = **6.6.6.6**)
- Substitute a fake gateway router
 - Intercept **all** of a host's off-subnet traffic
 - o (even if not preceded by a DNS lookup)
 - Relay contents back and forth between host and remote server
 - o **Modify** however attacker chooses
 - This is one type of invisible **Man In The Middle (MITM)**
 - o Victim host generally has no way of knowing it's happening! 😞
 - o (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

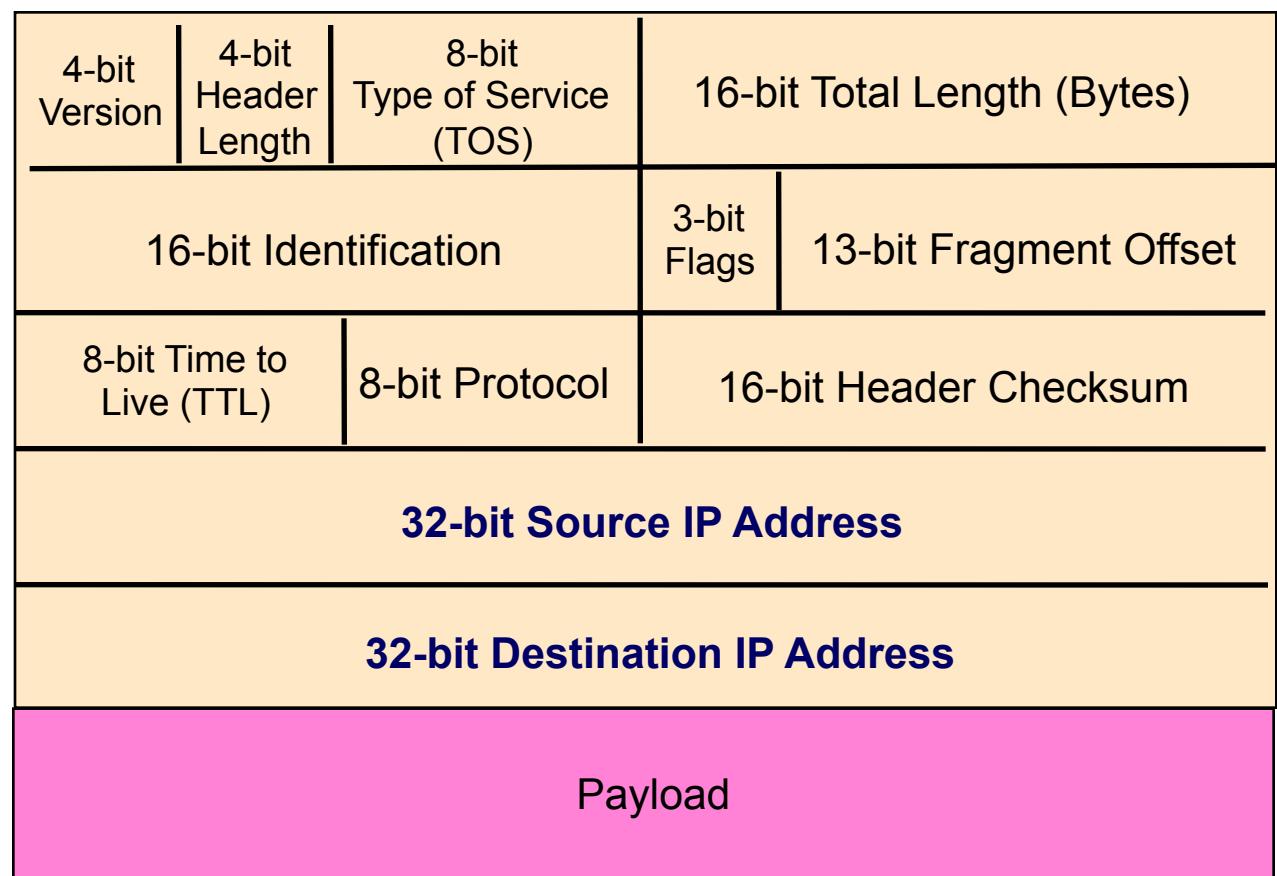
- DHCP threats highlight:
 - Broadcast protocols inherently at risk of **local** attacker spoofing
 - Attacker knows exactly when to try it ...
 - ... and can see the victim's messages
 - When initializing, systems are particularly vulnerable because they can *lack a trusted foundation* to build upon
 - 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



Bridges multiple “subnets” to provide *end-to-end* **internet** connectivity between **nodes**



IP = Internet Protocol

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)