CS489/689
Privacy, Cryptography, Network and Data Security

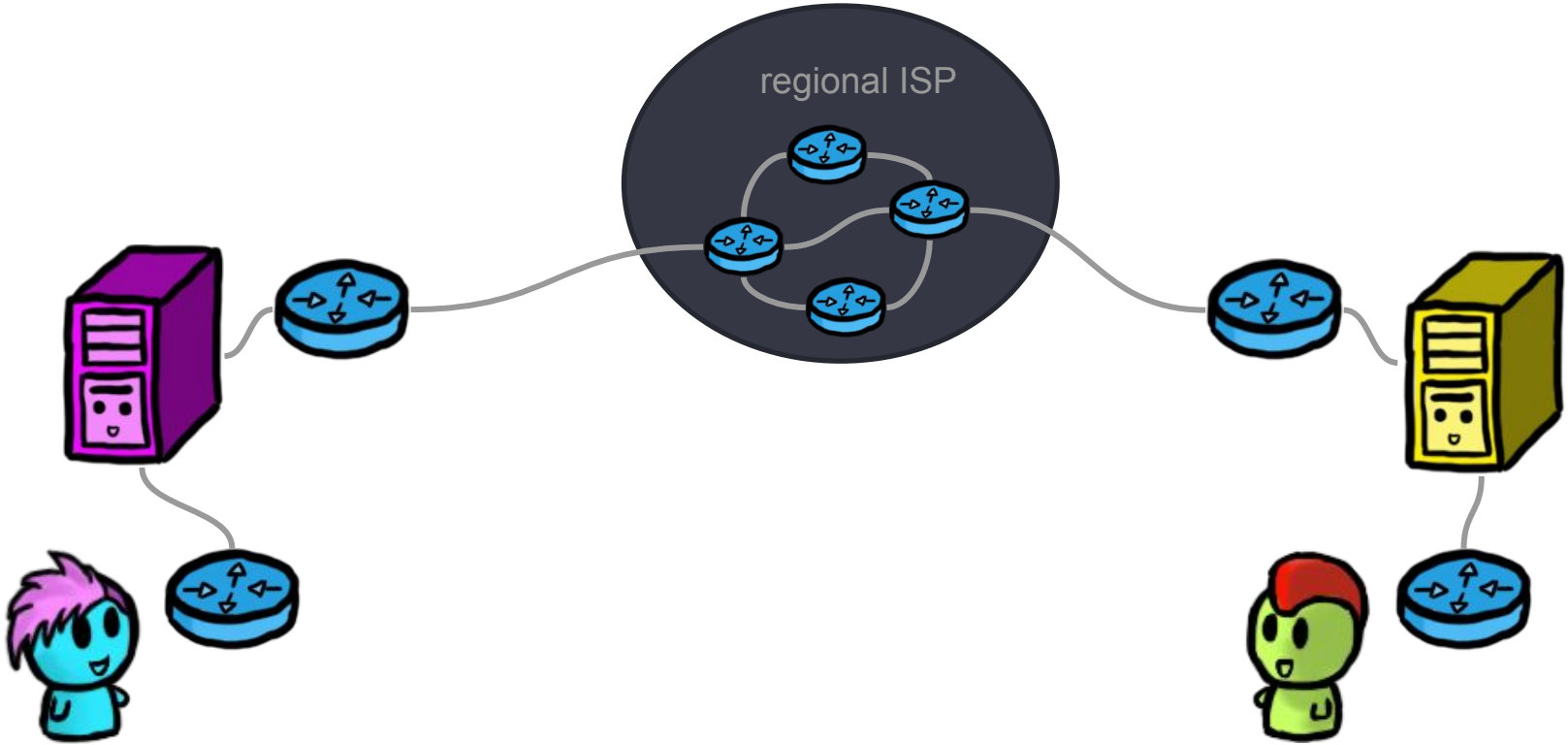
Winter 2023, Tuesday/Thursday 8:30-9:50am

Today: Authentication

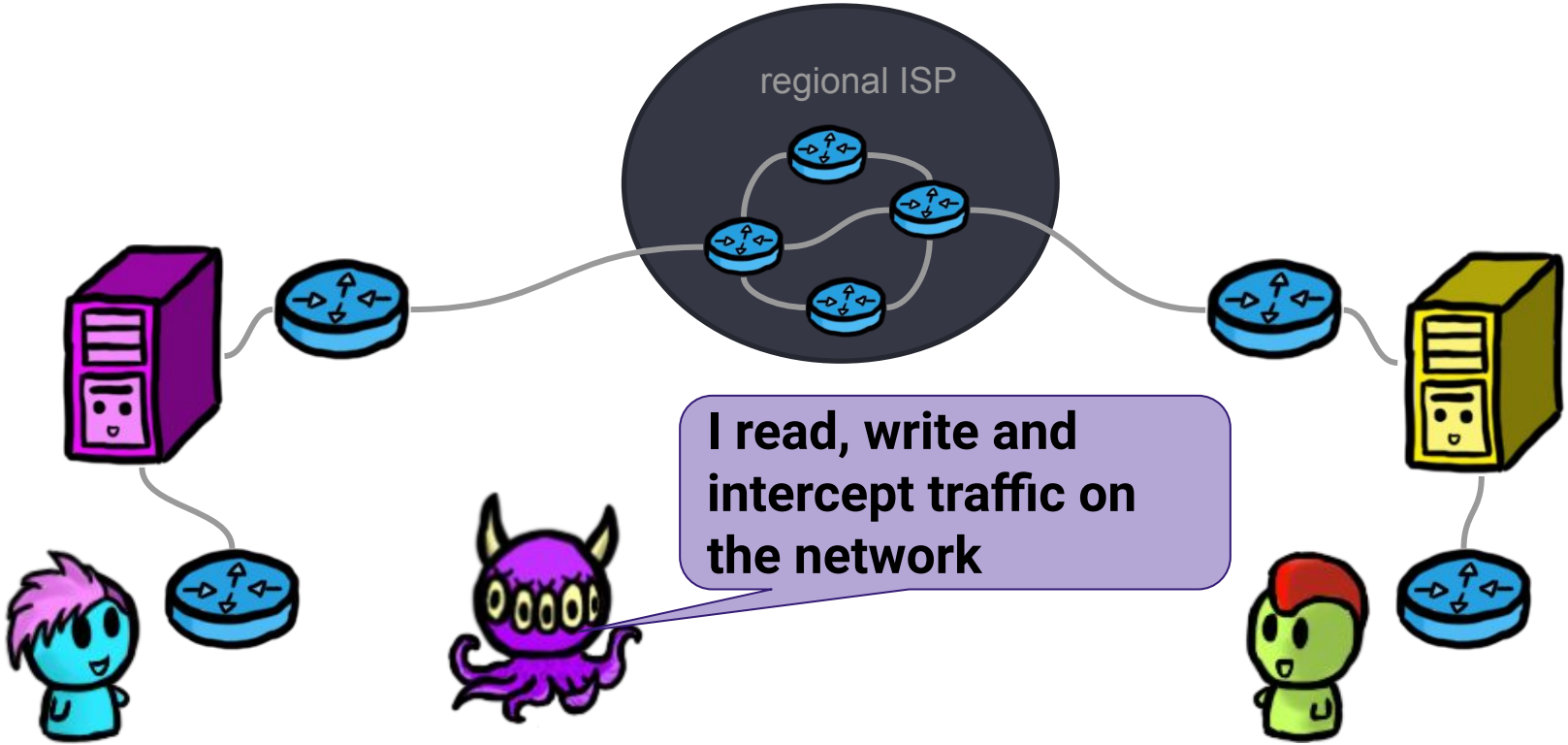
- Recap:

Authenticity: Prevent Mallory from **impersonating** Alice

Our Model: Who is talking to whom?

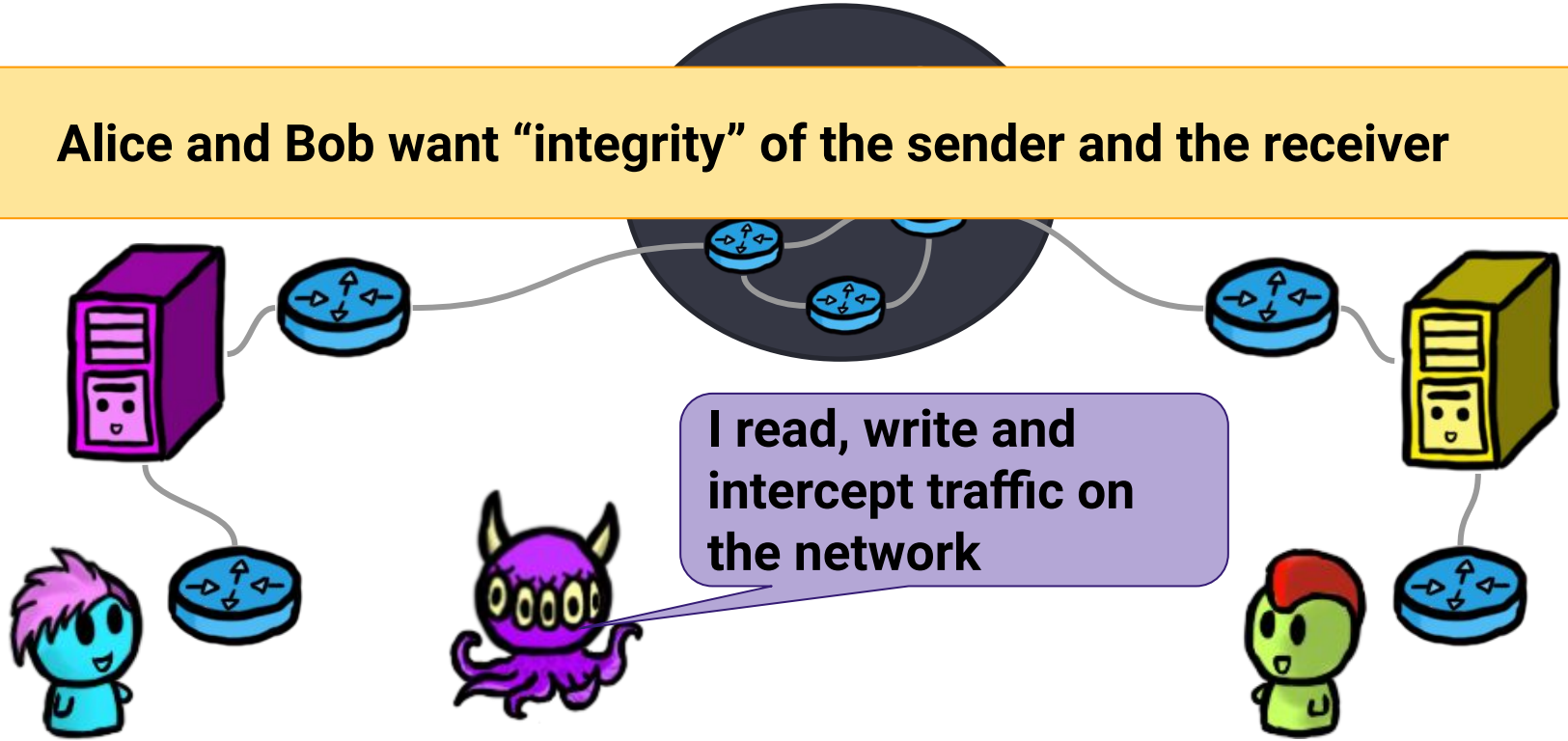


Our Model: Who is talking to whom?

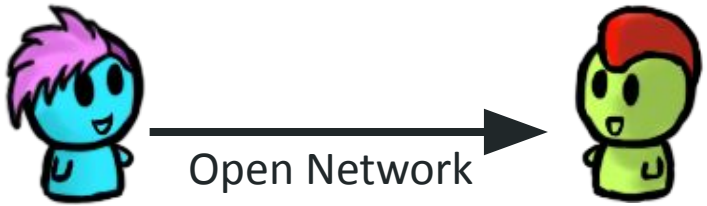


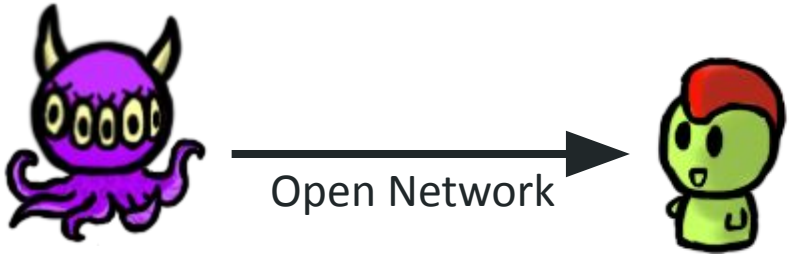
Our Model: Who is talking to whom?

Alice and Bob want “integrity” of the sender and the receiver



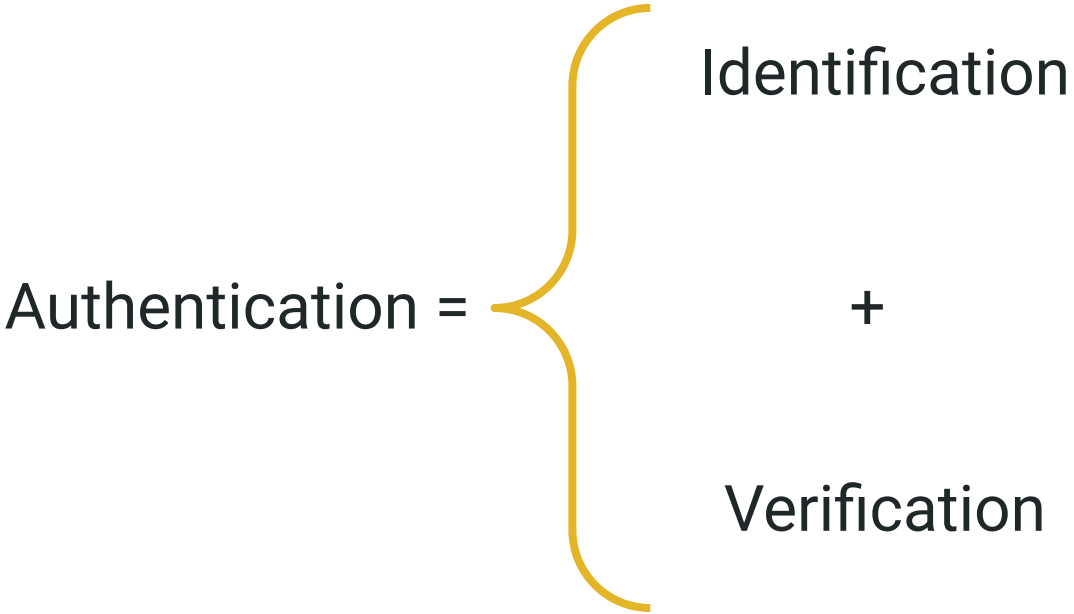
Our Model: Who is talking to whom?



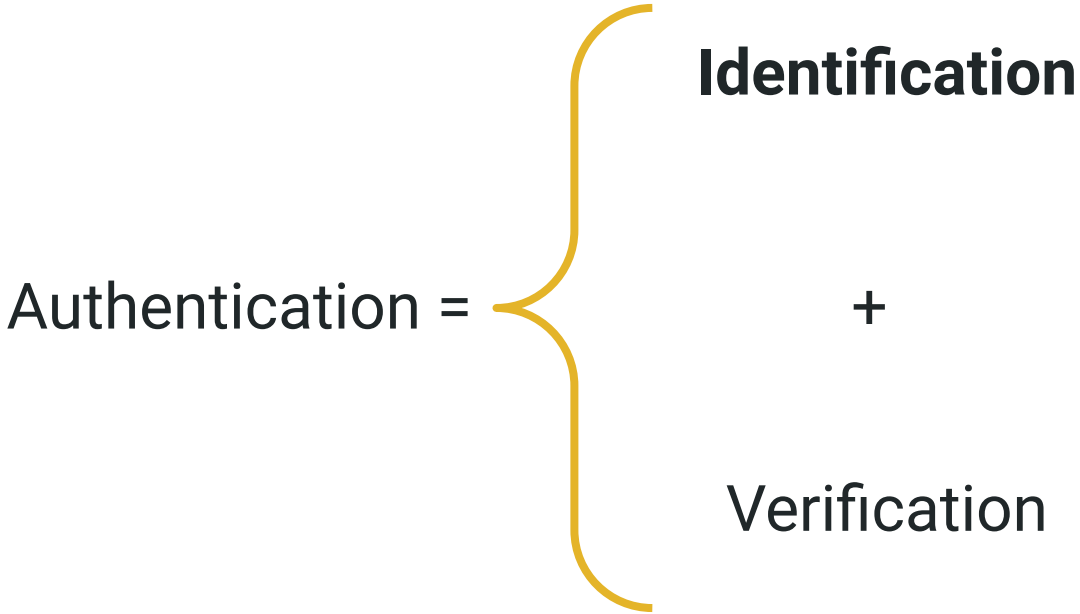


Goal: distinguish who you are talking to and confirm it

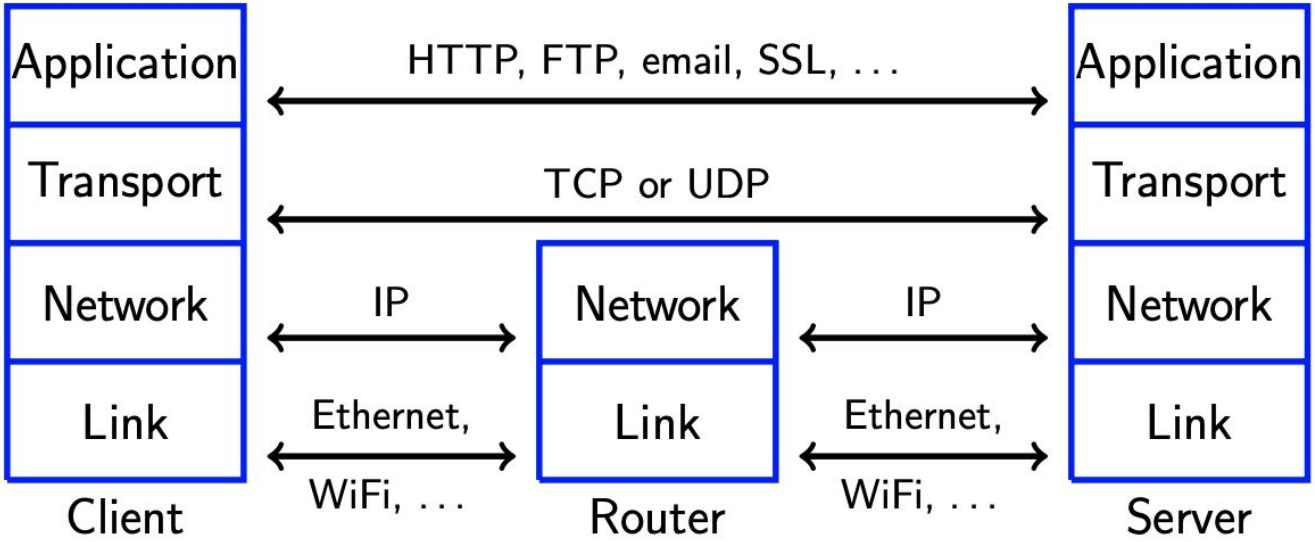
Definition of Authentication



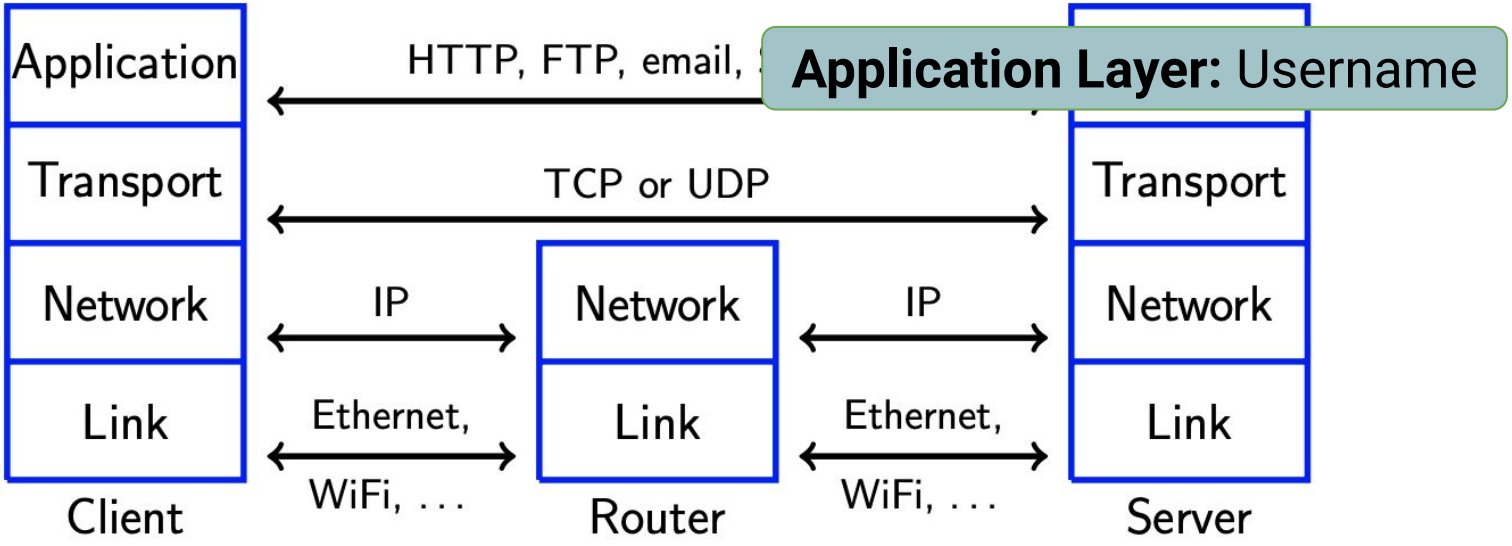
Definition of Authentication



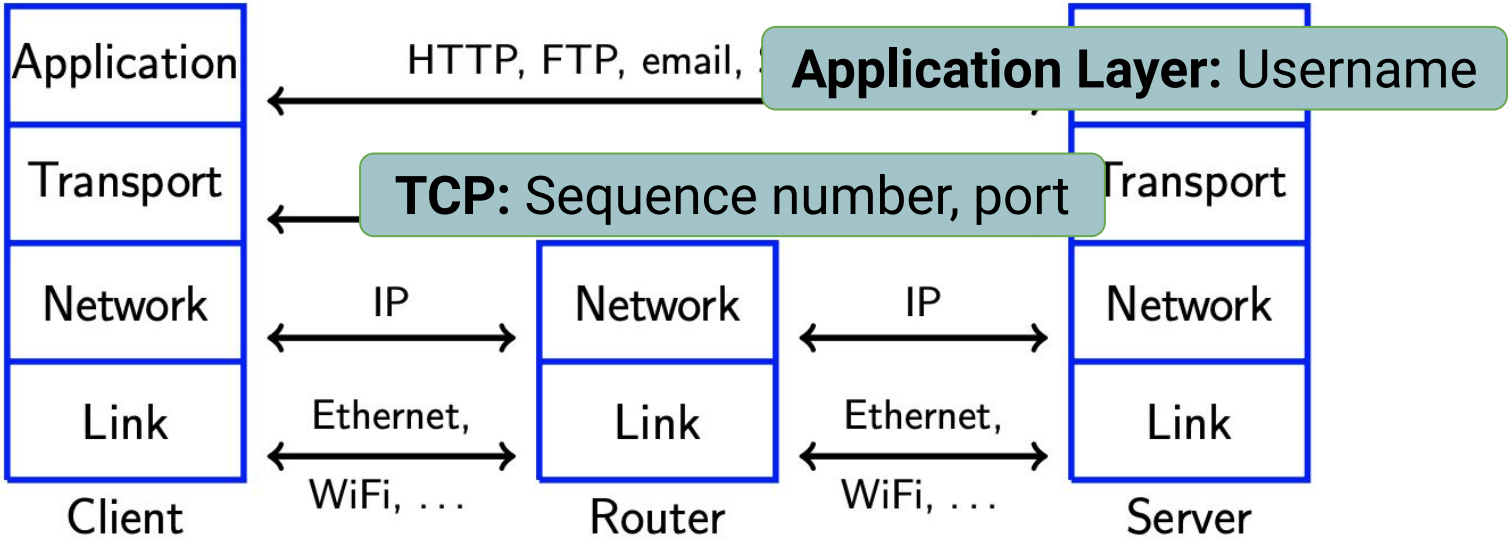
Identification on Network



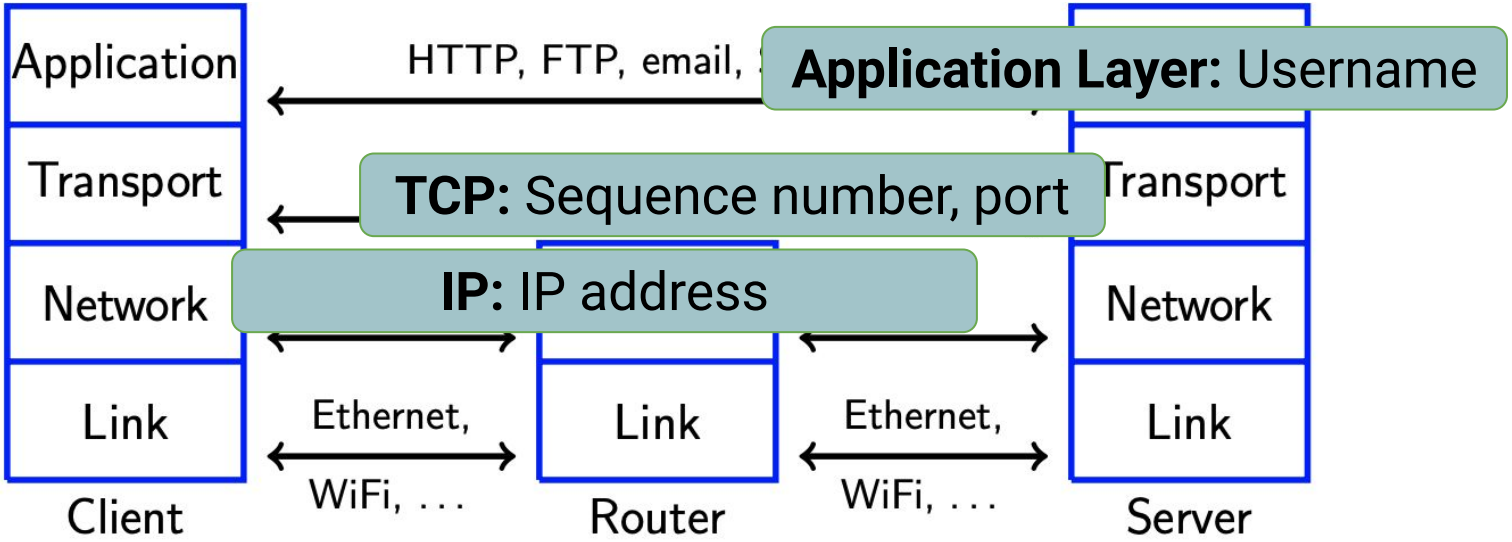
Identification on Network



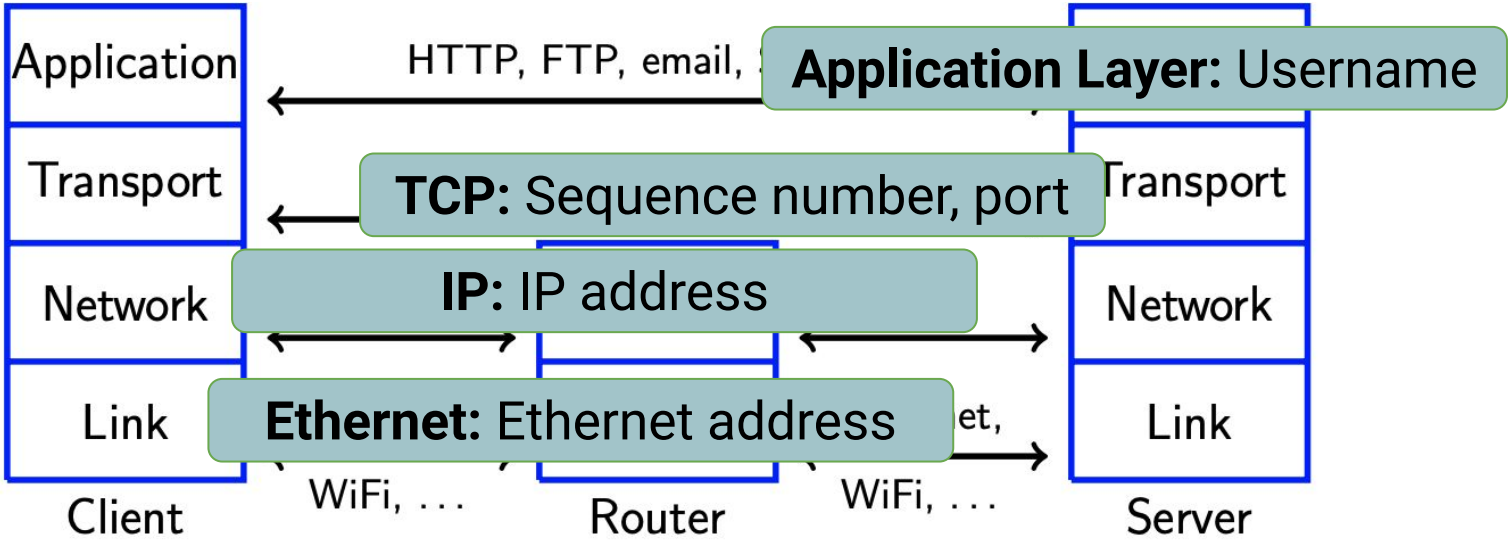
Identification on Network



Identification on Network



Identification on Network



Returning to Authentication

Authentication =

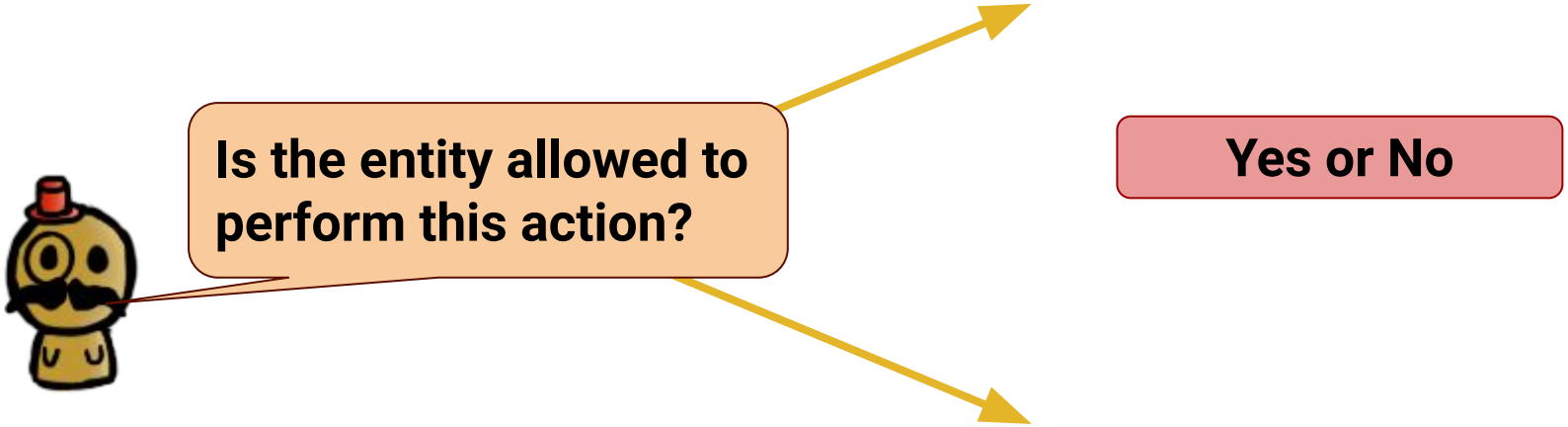
Identification

+

Verification

Need both: for example, to achieve access control

Access Control



Access Control

Is the entity allowed to perform this action?

Doesn't matter…unless you can verify the entities **identity**

Port Scan: **Identification** at the IP Layer

- Server offers services at different port (TCP state listening)

Port Scan: **Identification** at the IP Layer

- Server offers services at different port (TCP state listening)
- Client sends TCP SYN packet to all ports

Port Scan: **Identification** at the IP Layer

- Server offers services at different port (TCP state listening)
- Client sends TCP SYN packet to all ports
- If server is listening, then server responds with SYN+ACK packet

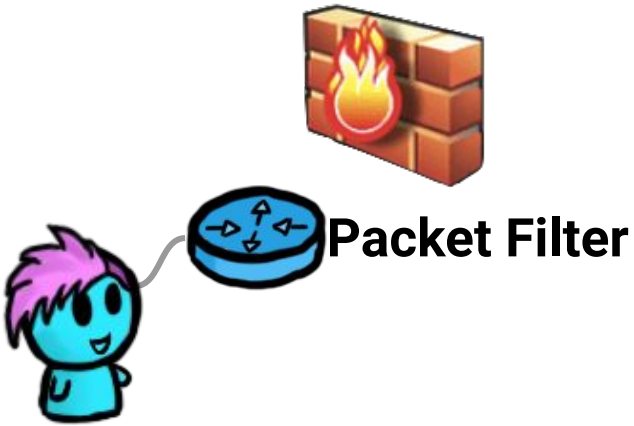
Port Scan: **Identification** at the IP Layer

- Server offers services at different port (TCP state listening)
- Client sends TCP SYN packet to all ports
- If server is listening, then server responds with SYN+ACK packet
- If server is not listening, then server responds with RST

Port Scan: **Identification** at the IP Layer

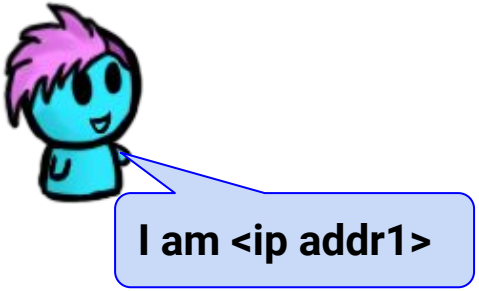
- Server offers services at different port (TCP state listening)
- Client sends TCP SYN packet to all ports
- If server is listening, then server responds with SYN+ACK packet
- If server is not listening, then server responds with RST
- Client learns services offered by server
  - Information for further attacks

Firewall (for Access Control)



- Only allows packets from IP address “A.B.C.D”
- Access control on source IP address (identification)
- IP address is not verified

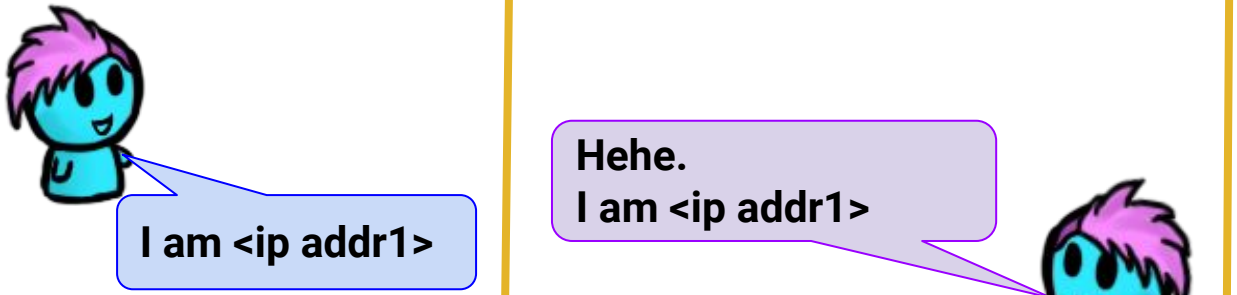
IP spoofing



Client's can set their source IP….

Try spoofs using attack simulator at: https://cs.uwaterloo.ca/~m2mazmud/netsim/

IP spoofing



Try spoofs using attack simulator at: https://cs.uwaterloo.ca/~m2mazmud/netsim/

IP spoofing

PROBLEM: Response packets routing for the spoofing/spoofed client

Try spoofs using attack simulator at: https://cs.uwaterloo.ca/~m2mazmud/netsim/

Reflected Port Scan: Spoofing + Ports

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Reflected Port Scan: Spoofing + Ports

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker pings A.B.C.D

Reflected Port Scan: Spoofing + Ports

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker pings A.B.C.D

A.B.C.D replies with increasing IP IDs (+1)

Reflected Port Scan: Spoofing + Ports

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker pings A.B.C.D

A.B.C.D replies with increasing IP IDs (+1)

Attacker spoofs SYN packet from A.B.C.D

Reflected Port Scan: Spoofing + Ports

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker pings A.B.C.D

A.B.C.D replies with increasing IP IDs (+1)

Attacker spoofs SYN packet from A.B.C.D

Server's response?

Reflected Port Scan: Spoofing + Ports Con't

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker spoofs SYN packet from A.B.C.D

Server responds RST to A.B.C.D

Reflected Port Scan: Spoofing + Ports Con't

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker spoofs SYN packet from A.B.C.D

Server responds RST to A.B.C.D 

Client (A.B.C.D) drops packet, IP id is not affected

Reflected Port Scan: Spoofing + Ports Con't

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker spoofs SYN packet from A.B.C.D

Server responds RST to A.B.C.D

Client (A.B.C.D) drops packet, IP id is not affected

Server responds SYN+ACK to A.B.C.D

Reflected Port Scan: Spoofing + Ports Con't

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker spoofs SYN packet from A.B.C.D

Server responds RST to A.B.C.D 

Client (A.B.C.D) drops packet, IP id is not affected

Server responds SYN+ACK to A.B.C.D 

A.B.C.D replies RST

Reflected Port Scan: Spoofing + Ports Con't

- Assume client uses incremental IP IDs (field in IP header used for fragmentation) e.g., $ID_{i+1} = ID_i + 1$

Attacker spoofs SYN packet from A.B.C.D

Server responds RST to A.B.C.D

Client (A.B.C.D) drops packet, IP id is not affected

Server responds SYN+ACK to A.B.C.D

A.B.C.D replies RST, so IP ID increases by 1

Attacker observes increases in ping replies

A Very Simple Public-Key Authentication Protocol

Client connects to the server and asks it to authenticate

Server reads a time T and sends a signature $Sign_S(T)$

Client reads a time T', verifies the signature and checks that T' is close to T

Attack 1: Adversary Authenticates as the Server

- Find an attack such that the adversary can authenticate as the server

Solution: Attack 1

- Mal connects to the server at time T and obtains $Sign_S(T)$

Solution: Attack 1

- Mal connects to the server at time T and obtains $Sign_S(T)$ 
- Client wants to connect to server at time T” 

Solution: Attack 1

- Mal connects to the server at time T and obtains $Sign_S(T)$ 
- Client wants to connect to server at time T” 
- Mal redirects request 

Solution: Attack 1

- Mal connects to the server at time T and obtains $Sign_S(T)$ 
- Client wants to connect to server at time T” 
- Mal redirects request 
- Mal manipulates the time at the client (e.g. Internet time protocol) to T’ 

Solution: Attack 1

- Mal connects to the server at time T and obtains $Sign_S(T)$ 
- Client wants to connect to server at time T” 
- Mal redirects request 
- Mal manipulates the time at the client (e.g. Internet time protocol) to T’ 
- Mal responds with $Sign_S(T)$ 
    - Replay attack

Solution: Attack 1

- Mal connects to the server at time T and obtains $Sign_S(T)$ 
- Client wants to connect to server at time T” 
- Mal redirects request 
- Mal manipulates the time at the client (e.g. Internet time protocol) to T’ 
- Mal responds with $Sign_S(T)$ 
  - Replay attack
- Client reads time T’, verifies signature and accepts 
- The information signed must be *fresh*

Simple Public-Key Authentication Protocol 2

Client sends a random challenge r

Server responds with signature $Sign_S(r)$

Client verifies signature

Attack 2: Adversary Authenticates as the Server

- Find an attack such that Ingrid can authenticate as the server S

Solution: Attack 2

- Client sends a random challenge r up server

Solution: Attack 2

- Client sends a random challenge r up server 
- Mal redirects request to themself 

Solution: Attack 2

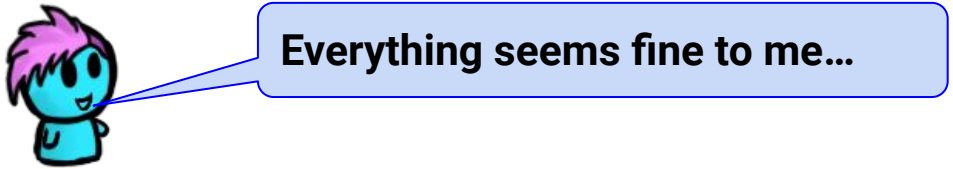
- Client sends a random challenge r up server 
- Mal redirects request to themself 
- Mal sends the request to server with ***same*** r 

Solution: Attack 2

- Client sends a random challenge r up server 
- Mal redirects request to themself 
- Mal sends the request to server with ***same*** r 
- Server responds $Sign_s(r)$ to Mal 

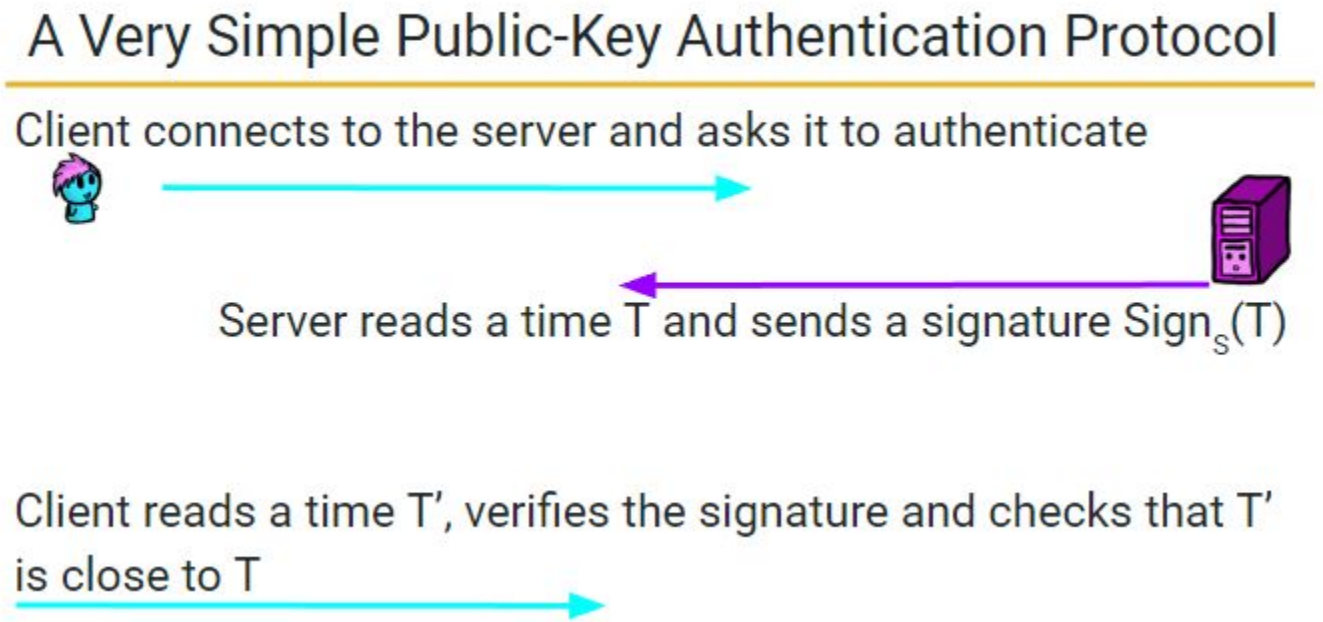
Solution: Attack 2

- Client sends a random challenge r up server 
- Mal redirects request to themself 
- Mal sends the request to server with ***same*** r 
- Server responds $Sign_s(r)$ to Mal 
- Mal forwards response to client 
- Client accepts



Question

- Provide a fix to the simple authentication protocol (first one)



Solution

- The server signs the message S|C|r
- Then Ingrid cannot forward S|I|r to the client and have it accept
- The problem existed in an early version SSL (now TLS)
- Challenge information needs to be *complete* (including communicating entities)

Verification, in a few slides

Verification Methods

- Something you know

Verification Methods

- Something you know
  - Password
- Something you have

Verification Methods

- Something you know
  - Password
- Something you have
  - Mobile Phone
  - Cryptographic Key
- Something you are
  - Biometric

Verification Methods

- Something you know
  - Password
- Something you have
  - Mobile Phone
  - Cryptographic Key
- Something you are
  - Biometric
- 2-factor authentication
  - Using two of the above

Curious about some cool research in this space? Look up “Shatter Secrets”

Verification Setup

- Verification requires trusted setup phase
  - Attacker cannot modify the authentication information delivered
    - Mallory-in-the-middle attack
  - Identity can be established
- In a distributed system this implies a secure channel

Authentication Information Needs to Be Protected

- Password
  - Hashed with Salt
- Public Key
  - Doesn’t allow inference of private key
- Biometric Template
  - Open Problem (Crypto?)

No Verification **does not imply** Anonymity (No ID)

- Implicit identifiers
  - IP address
    - Your Internet provider knows your IP address
  - Browser fingerprint
    - Fonts, browser capabilities (JavaScript, etc.), …
  - Web Cookies
  - Behavior
    - Typing, Walking, …
  - Location (Trajectory)
- Communication parties can identify each other without explicit identification
  - Servers can track your browser fingerprint (cookies)

Web Cookies

- Set in the HTTP protocol and stored on the browser
  - Session vs. permanent
- Stored cookies are automatically transferred on each request to the same domain
- Used for authentication
- Used for tracking
  - Third-party cookies
    - Cookies set for different domains (option in HTTP protocol)
    - Cookies set by loaded objects (JavaScript, Images, etc.)

Recommended: Web tracking activity

- Launch the Labtainer VM
- In the terminal type: **`labtainer webtrack`** and hit enter
- Open the lab manual
- Complete the lab

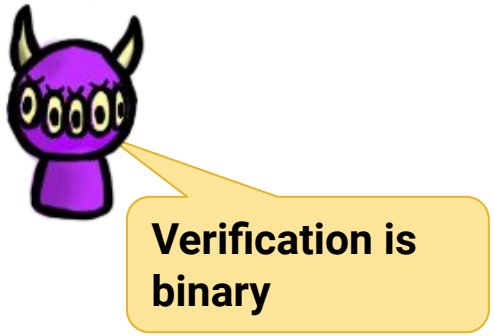
Verification, what’s the catch?

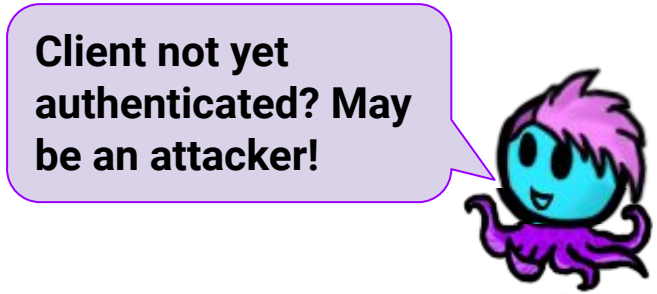
Verification: May Imply Leakage

Verification is binary

Yes or No

Verification: May Imply Leakage





Yes or No

Verification: May Imply Leakage

Verification is binary

Verification is given to the client…which could be me.

Client not yet authenticated? May be an attacker!

Verification attempts can leak information.

Yes or No

Verification: May Imply Leakage

Verification is binary

Verification is given to the client…which could be me.

Client not yet authenticated? May be an attacker!

Verification attempts can leak information.

Q: Consider a failed login attempt. What could it reveal?

Yes or No

Verification: May Imply Leakage

Verification is binary

Verification is given to the client…which could be me.

Verification attempts can leak information.

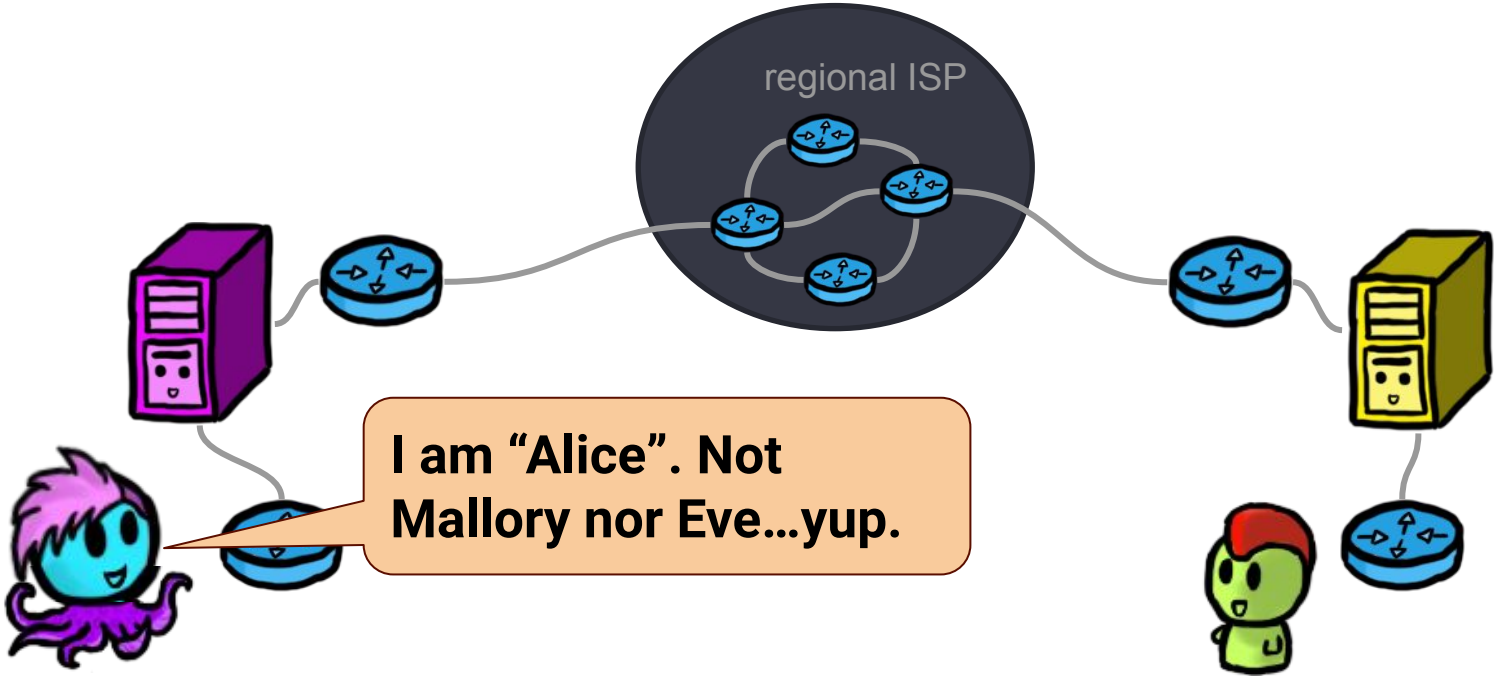
Client not yet

Q: Consider a failed login attempt. What could it reveal?

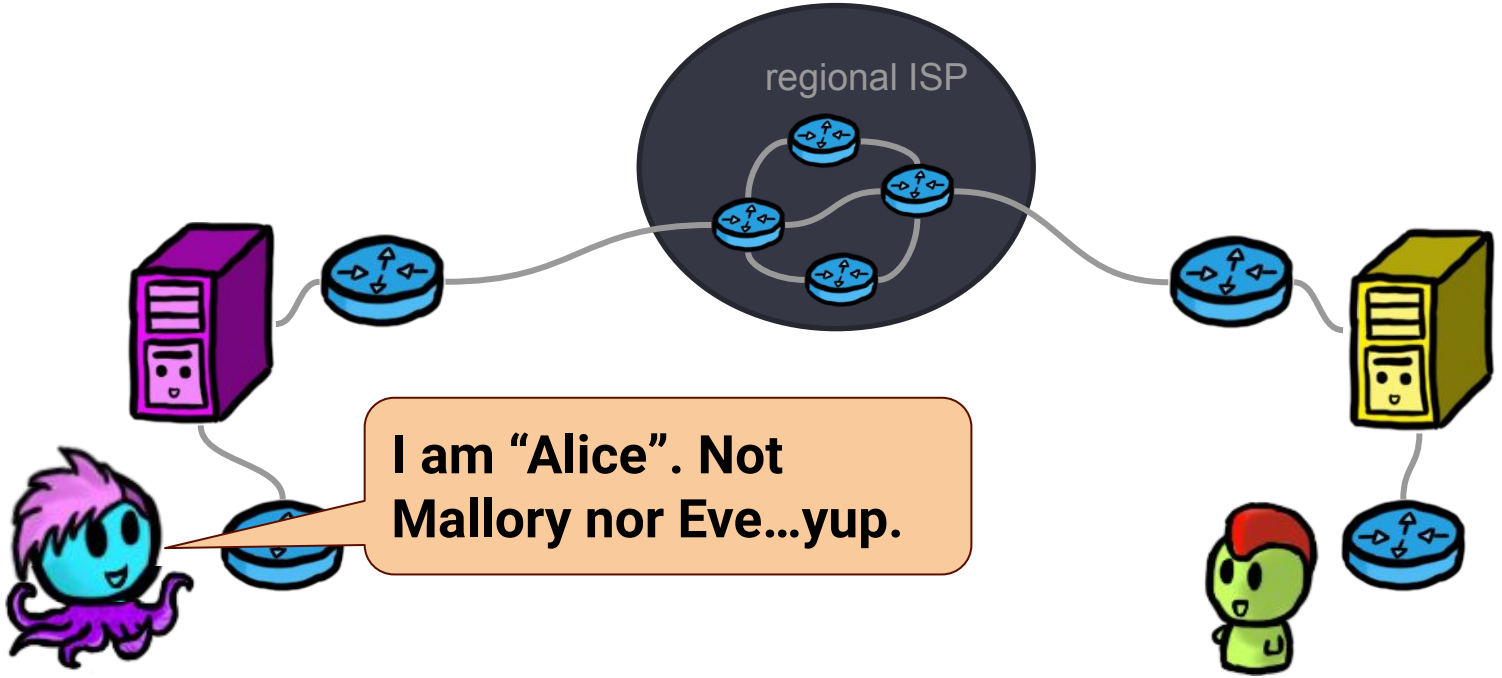
Yes or No

A: wrong user name, wrong password

Verification may be abused



Verification may be abused



Identification/Authentication information may be supplied by attacker

Impersonation attacks go both ways…

- Client
  - MAC spoofing
  - IP spoofing
  - Session hijacking
  - Guessed password login
  - Spam

Impersonation attacks go both ways…

- Client
  - MAC spoofing
  - IP spoofing
  - Session hijacking
  - Guessed password login
  - Spam
- Server
  - Broadcast networks (Ethernet bridge poisoning)
  - Rerouting attacks (ARP, ICMP redirect, RIP/BGP)
  - DNS cache poisoning
  - Phishing

DNS cache poisoning

- Victim and Attacker share common DNS server (cache)

DNS cache poisoning

- Victim and Attacker share common DNS server (cache)
- Attacker queries my.attack.home
  - Attacker colluded with my.attack.home authoritative DNS server
  - Attacker queries its DNS server for my.attack.home
  - my.attack.home DNS server replies with
    - my.attack.home A.B.C.D
    - www.yourbank.ca A.B.C.E
  - Victim queries its DNS server for www.yourbank.ca
  - DNS server replies from cache with A.B.C.E
  - Victim communicates with A.B.C.E

Denial of service

- Client requests more resources than server / network can deliver

Denial of service

- Client requests more resources than server / network can deliver
- Can be distributed (DDOS)
  - Multiple clients collude

Denial of service

- Client requests more resources than server / network can deliver
- Can be distributed (DDOS)
  - Multiple clients collude
- Examples
  - IP bandwidth: Ping flood
  - IP state: fragmentation attacks, routing attacks (blackhole)
  - TCP state: SYN flood, hash table attacks on stateful firewalls
  - TCP/IP Implementation error: Smurf, XMas, Ping of death, …

Attempts at Retrofitting Authentication

Challenge: Resource Allocation in Networks

- Difficult due to distributed nature

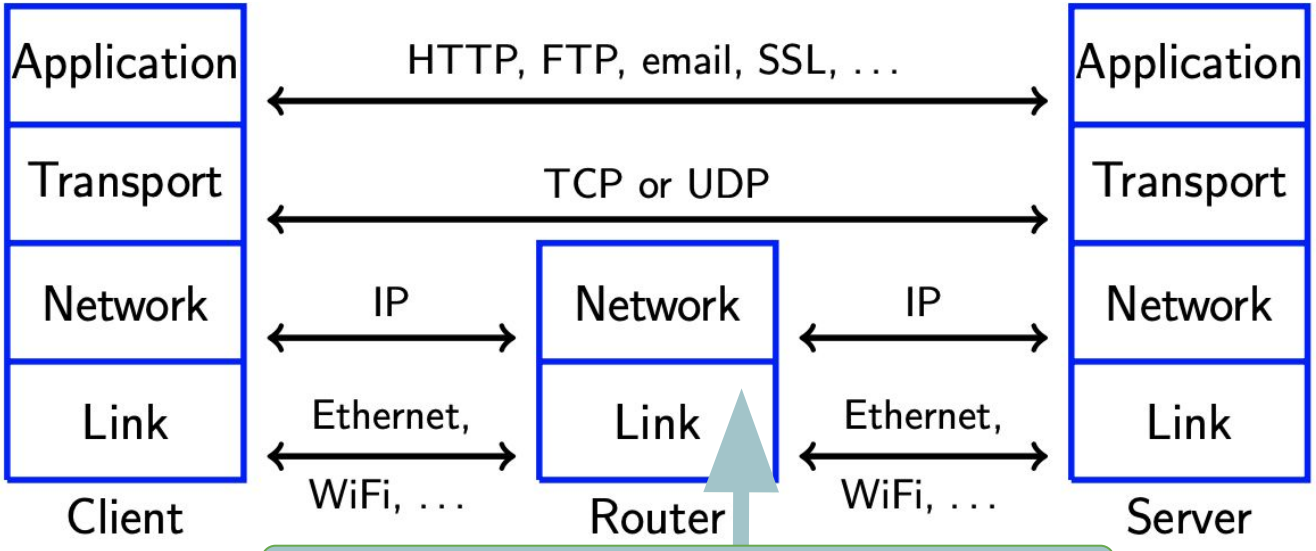
Challenge: Resource Allocation in Networks

- Difficult due to distributed nature
- Often no authentication of clients
  - Resource allocation can be foiled

Challenge: Resource Allocation in Networks

- Difficult due to distributed nature
- Often no authentication of clients
  - Resource allocation can be foiled
- Clients can be remote controlled / abused
  - Botnet (Storm, Mirai)
  - Reflectors (Ping with spoofed source)
  - Amplifiers (SNMP)

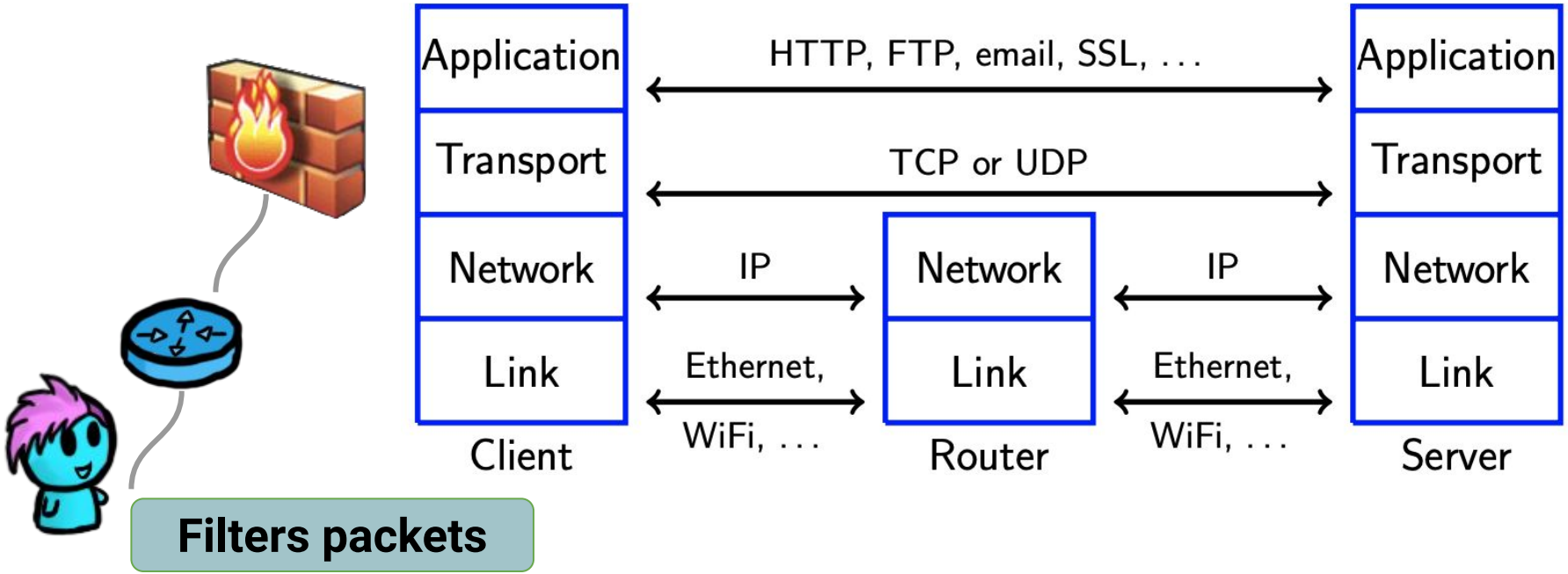
Retrofitting Authentication: WPA2



Wi-Fi Protected Access (Feb 14)

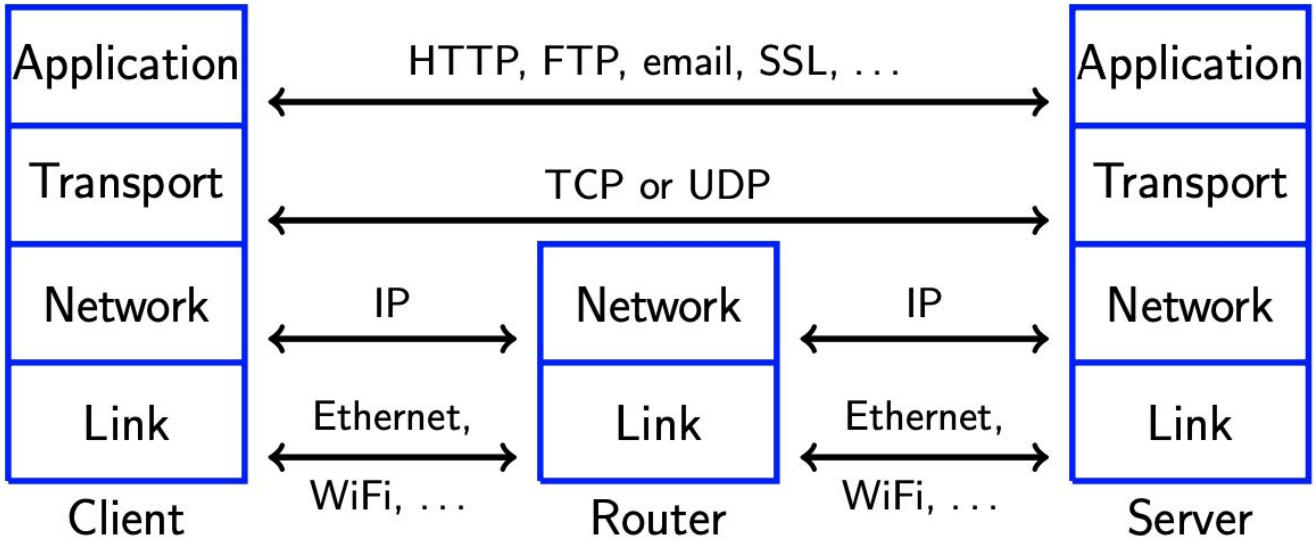
Cryptographic protection at the wireless data link layer

Retrofitting Authentication: Egress Filtering



Firewall at source verifying source IP

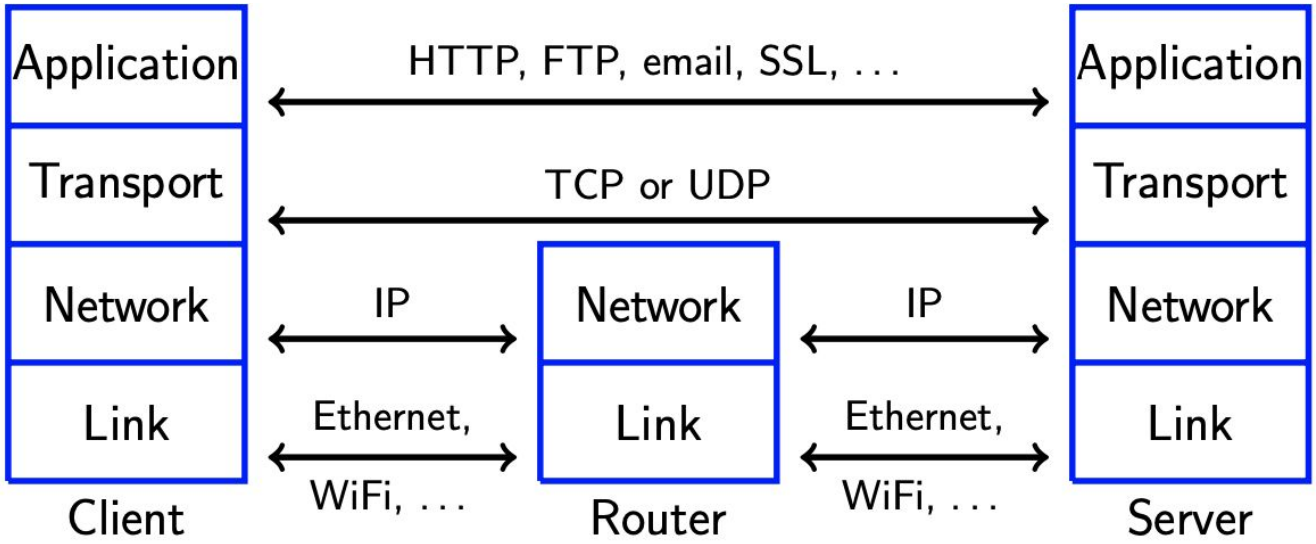
Retrofitting Authentication: IPSEC



Feb 7 (ish)

Cryptographic protection (MAC, symmetric encryption) at IP layer

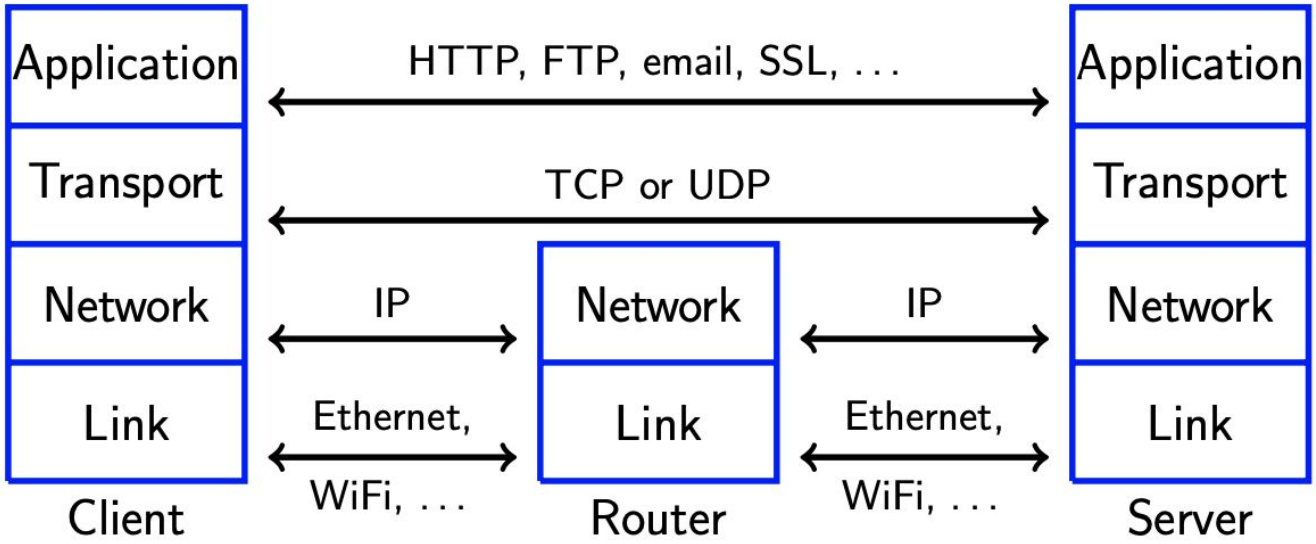
Retrofitting Authentication: DNSSEC



Feb 7

Cryptographic protection (Signature of DNS records) at DNS layer

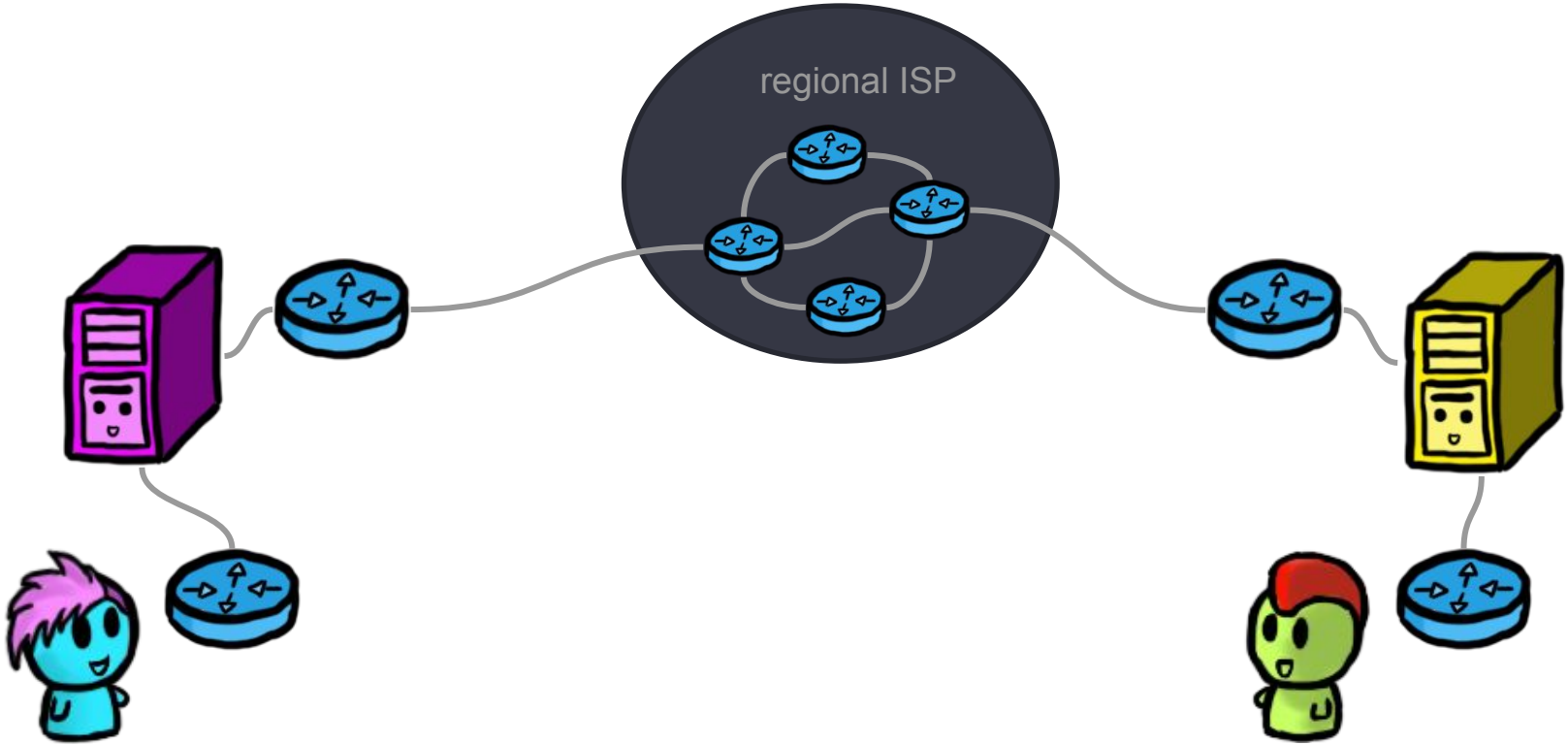
Retrofitting Authentication: TLS



Feb 14

Cryptographic protection at session (between TCP and application) layer

So now what? Actual Protocols



Next: NetSec continues…

Exercise

- Fully homomorphic encryption allows to compute any function f over encrypted data, i.e., f(E(x)) = E(f(x))
- Why does that **not** solve the biometric template protection?