Lec 13: Secure Communication

CSED415: Computer Security

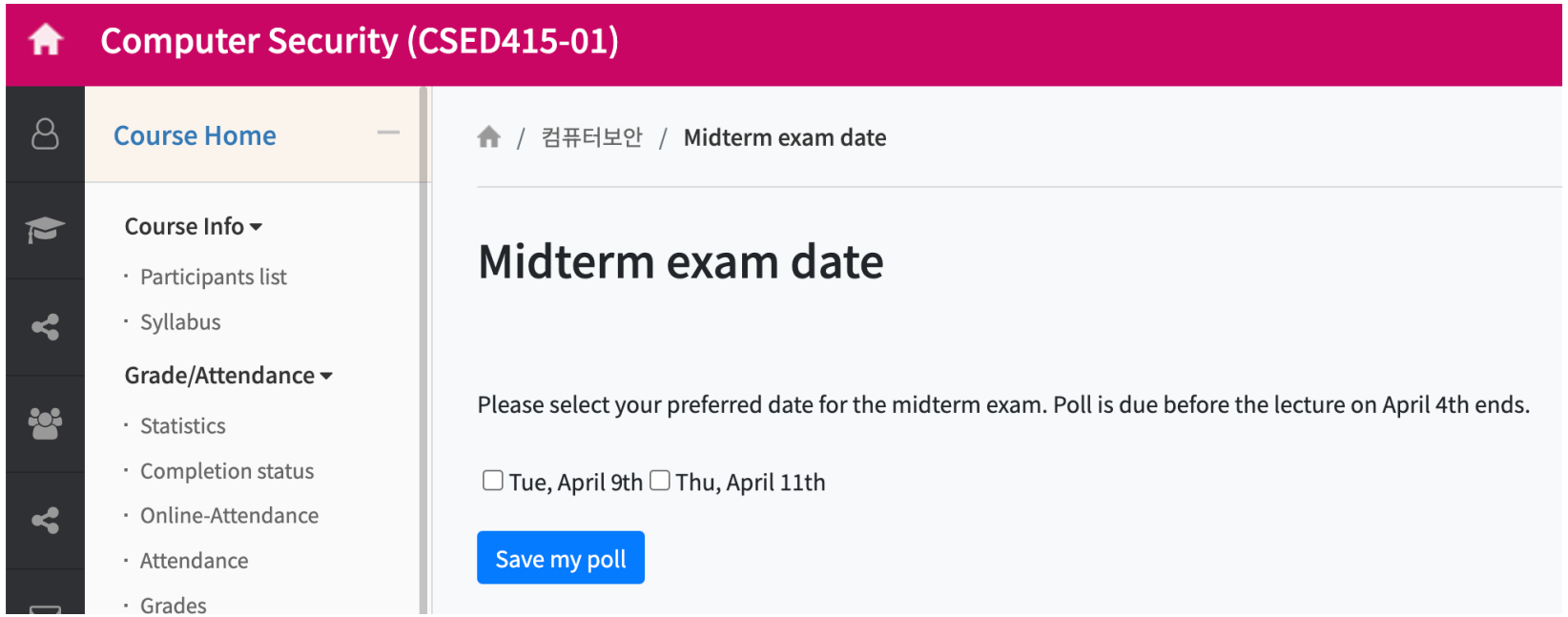
Spring 2024

Seulbae Kim

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Administrivia

- Please vote for Midterm exam date!
  - Poll on PLMS: Select April 9th or 11th
  - Poll is due before the lecture on April 4th ends



Recap

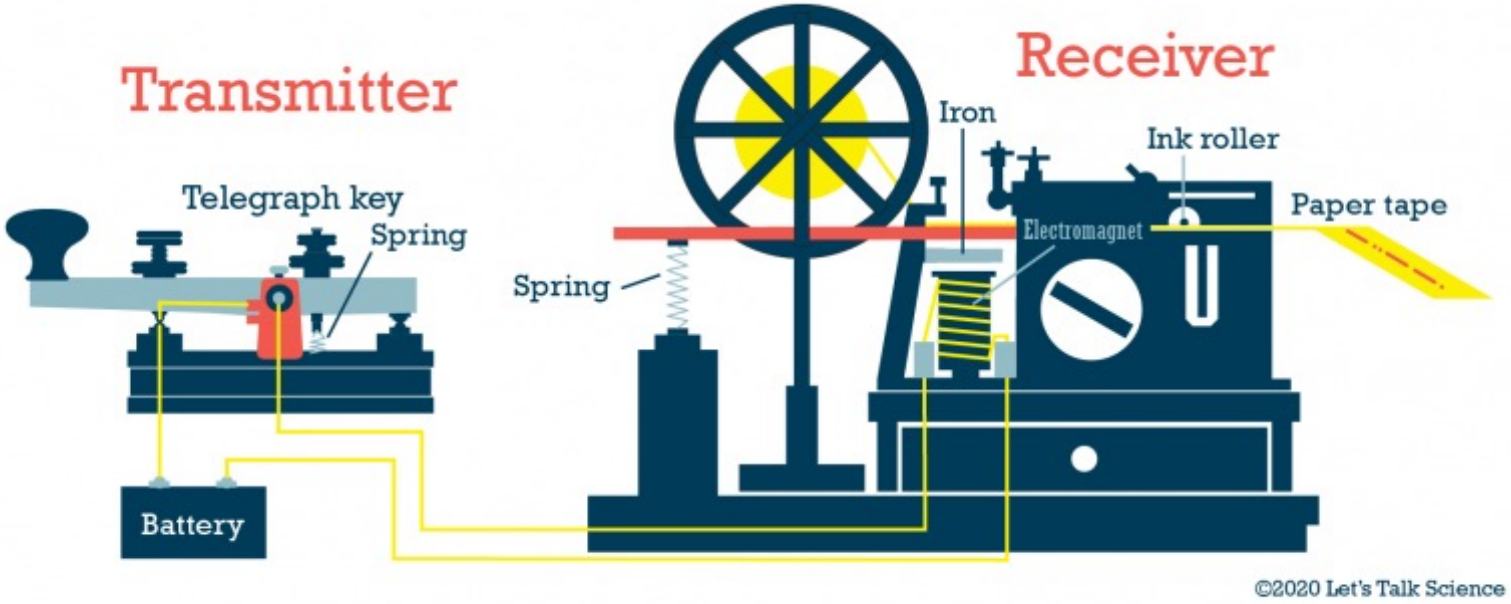
- Cryptography:
  - A means for secure communication over insecure channels
- Many network systems utilize cryptography for secure communication
  - How does various internet services utilize cryptographic primitives to ensure secure connection in practice?

Secure Emails

Brief history of email

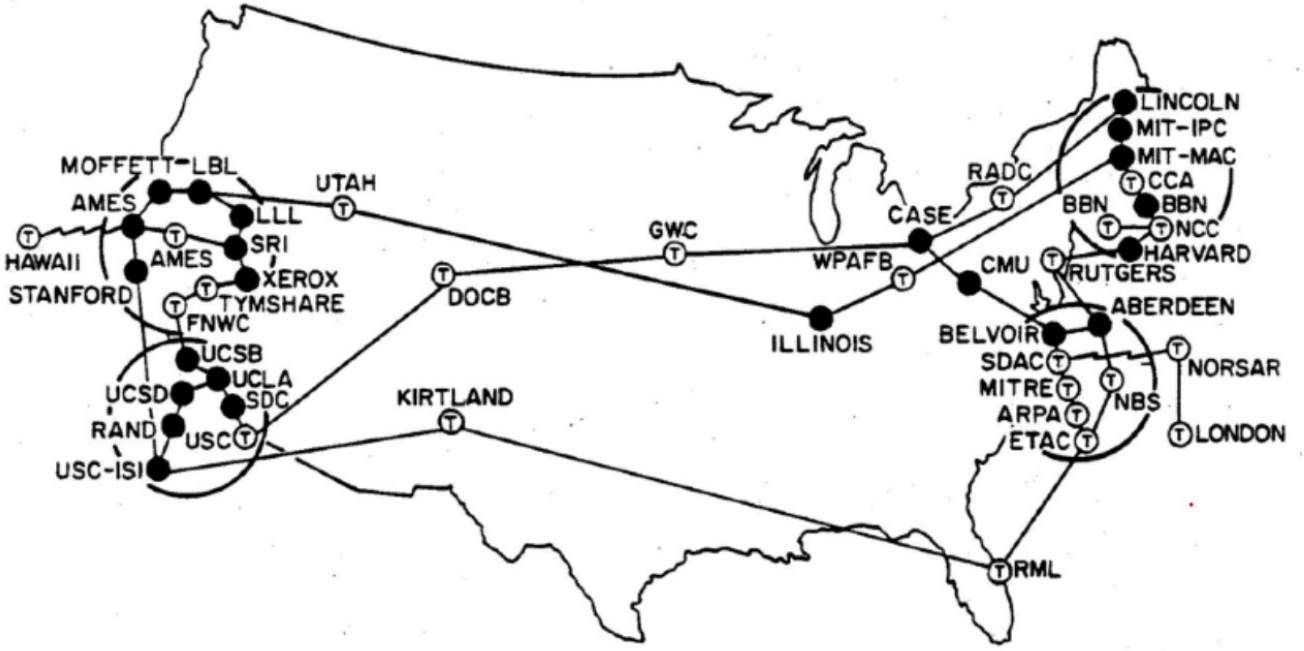
- Physical transportation

- Electrical telegraphs with morse code (1800s)



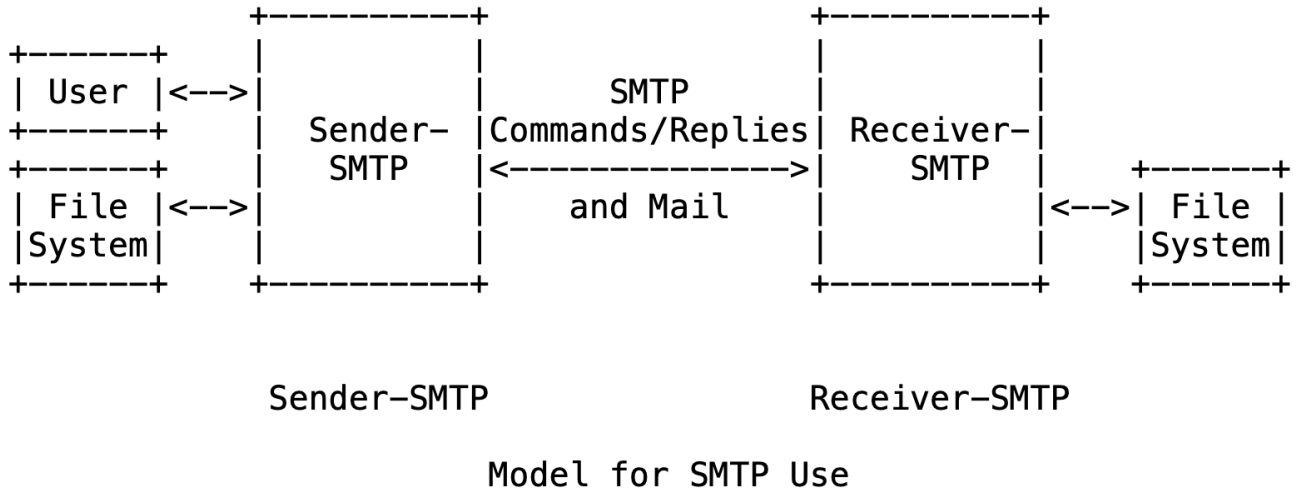
Brief history of email

- ARPANET by US Department of Defense (1971)
  - The first email sent via SNDMSG program on ARPANET
  - Used @ symbol to designate recipient computer's address



Brief history of email

- SMTP (Simple Mail Transfer Protocol, 1980)
  - Standardized protocol for email transmission
  - Simple messages (e.g., plaintexts) can be sent using SMTP



Img: RFC 821. Simple Mail Transfer Protocol

Brief history of email

- Multipurpose Internet Mail Extensions (MIME, 1991)
  - Extends the format of email messages to support multimedia
  - MIME header contain fields about the e-mail body
    - MIME-Version
    - Content-Type
      - Text/plain, image/jpeg, audio/mp3, …
    - Content-Disposition
      - Inline, attachment
    - Content-Transfer-Encoding
      - base64, ascii, …

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary=frontier
This is a message with multiple parts in MIME format. --frontier

Content-Type: text/plain
This is the body of the message. --frontier

Content-Type: application/octet-stream
Content-Transfer-Encoding: base64
PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA+
VGhpcyBpcyB0aGUg Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9
ib2R5Pgo8L2h0bWw+Cg==

--frontier--
```

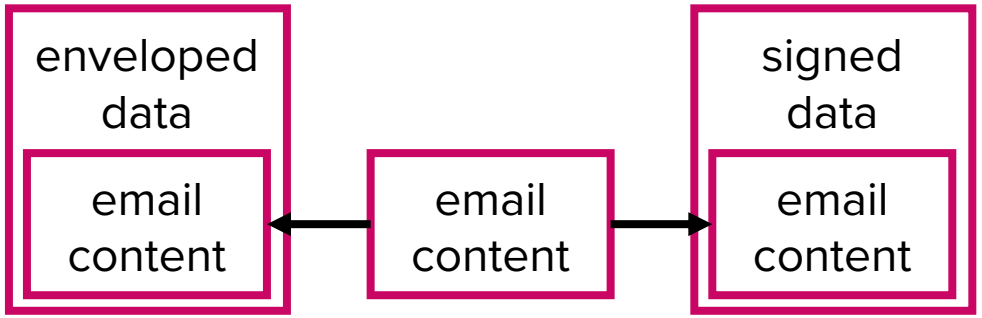
No notion of security so far

S/MIME: Secure MIME

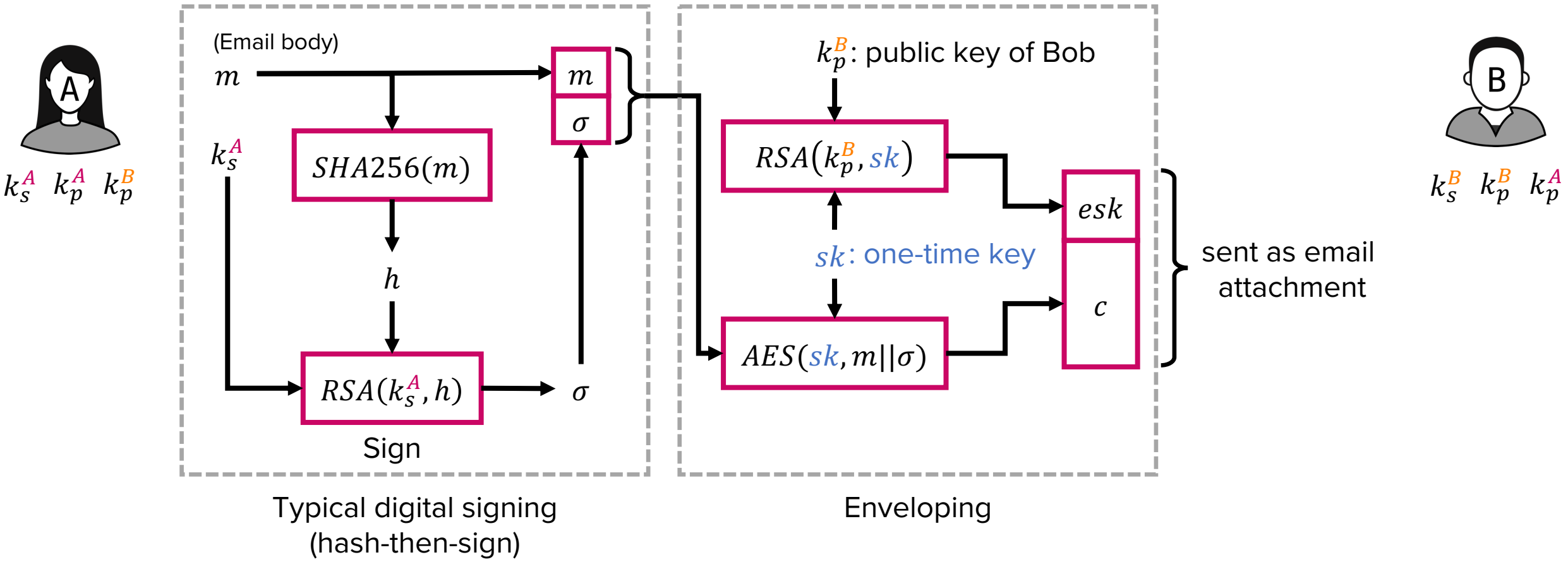
- Defined as a set of additional MIME content types
  - e.g., Content-Type: application/pkcs7-mime

(PKCS: public-key cryptography standards)

- Provide the ability to sign and/or encrypt email messages
  - Enveloped data: encrypted message
  - Signed data: plain message + digital signature of the message
  - Signed and enveloped data: digitally signed and encrypted message



S/MIME functional flow



S/MIME email example

POSTECH

```
MIME-Version: 1.0
Message-Id: <9358910051929015@postech.ac.kr>
Date: Tue, 02 Apr 2024 00:16:31 +0900 (Korea Standard Time)
From: alice@postech.ac.kr
To: bob@postech.ac.kr
Subject: email example
Content-Type: application/pkcs7-mime; name=smime.p7m; smime-type=enveloped-data
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

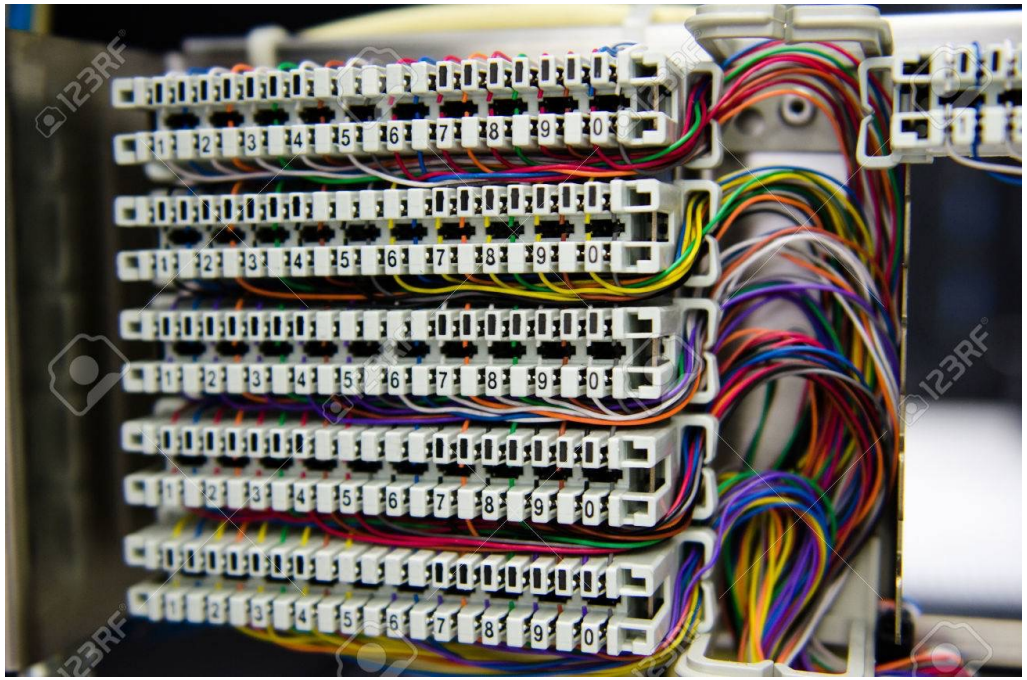
```
MIIBHgYJKoZIhvcNAQcDoIIBDzCCAQsCAQAxgcAwgb0CAQAwJjASMRAwDgYDVQQDEwdDYXJ
sUlNBAhBGNGvHgABWvBHTbi7NXXHQMA0GCSqGSIb3DQEBAQUABIGAC3EN5nGIiJi2lsGPcP
2iJ97a4e8kbKQz36zg6...bGgzoyEd8Ch4H/dd9gdzTd+taTEgS0ipdSJuNnkVY4/M652jK
LFf02hosdR8wQwYJKoZIhvcNAQcBMBQGCCqGSIb3DQMHBAgtaMXpRwZRNYAgDsiSf8Z9P43
LrY4OxUk660cu1lXeCSFOSOpOJ7FuVyU=
```

base64-encoded $esk + c$ attachment

Secure Socket Layer (SSL) / Transport Layer Security (TLS)

Background: Computer network

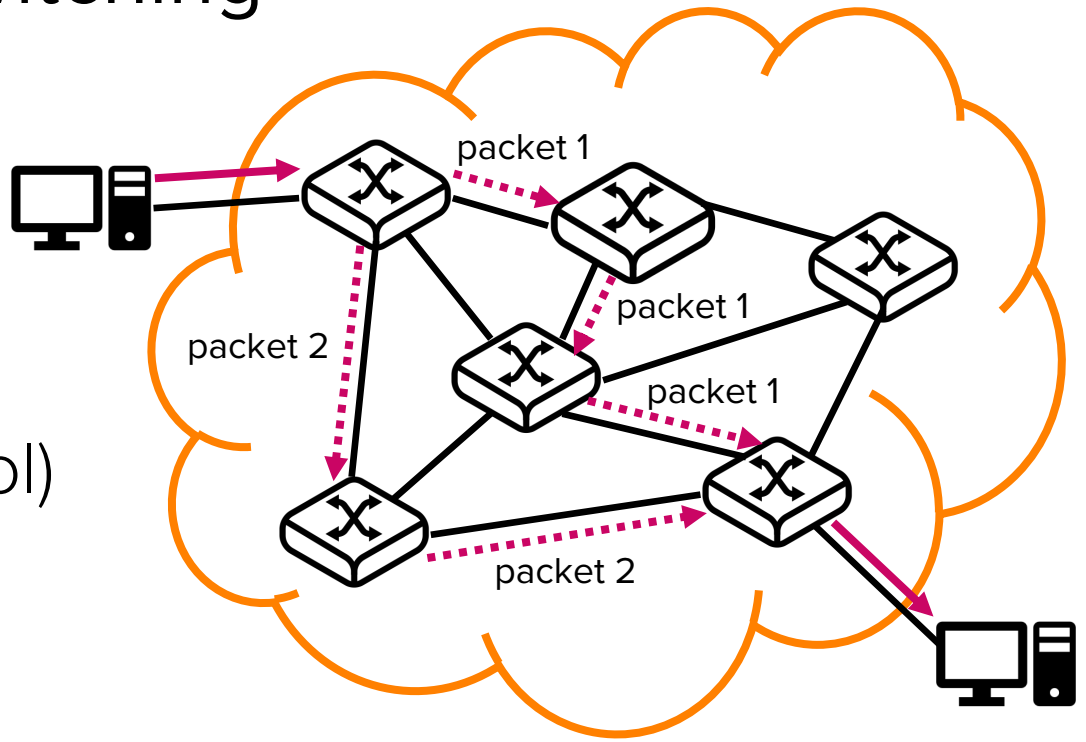
- Remote communication before internet: Circuit switching
  - Legacy phone network
  - Single route through sequence of hardware devices is established when two nodes communicate
  - Data (electric current) is sent along the route
  - The route is maintained until the communication ends



Telephone switchboard

Background: Computer network

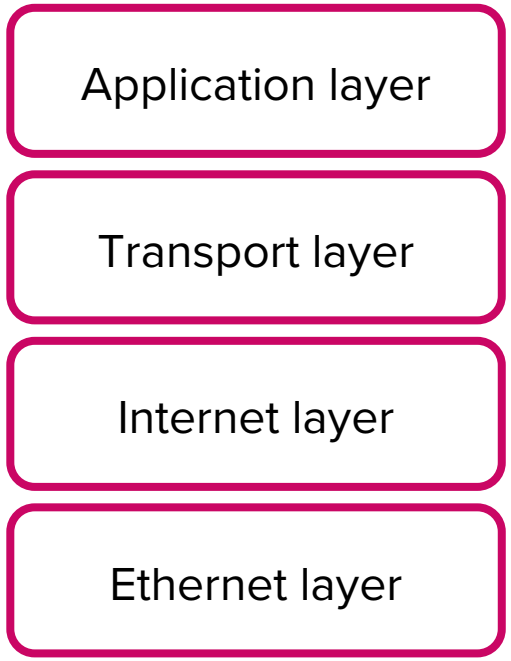
- Internet communication: Packet switching
  - Data is split into packets
  - Packets are transported independently through network
  - Network switches determine the best route for each packet (routing protocol)
  - Consequently, packets may follow different routes



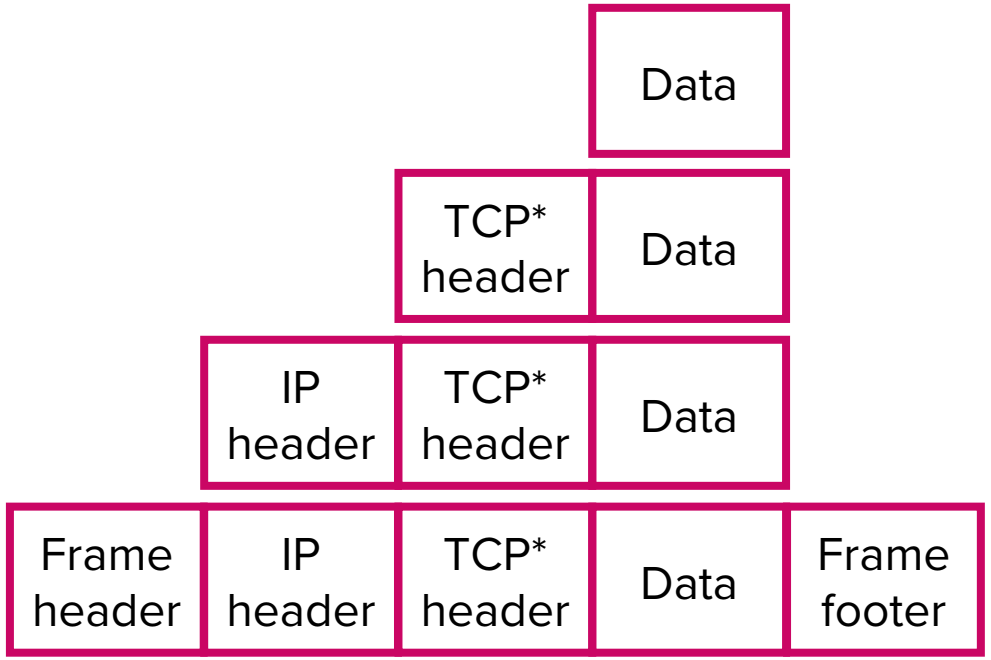
Packets sent over routers

Background: Computer network

- Computer networking utilizes a stack of layers
  - Higher layers use the services of lower layers via encapsulation



TCP/IP network model



Data encapsulation

*or UDP

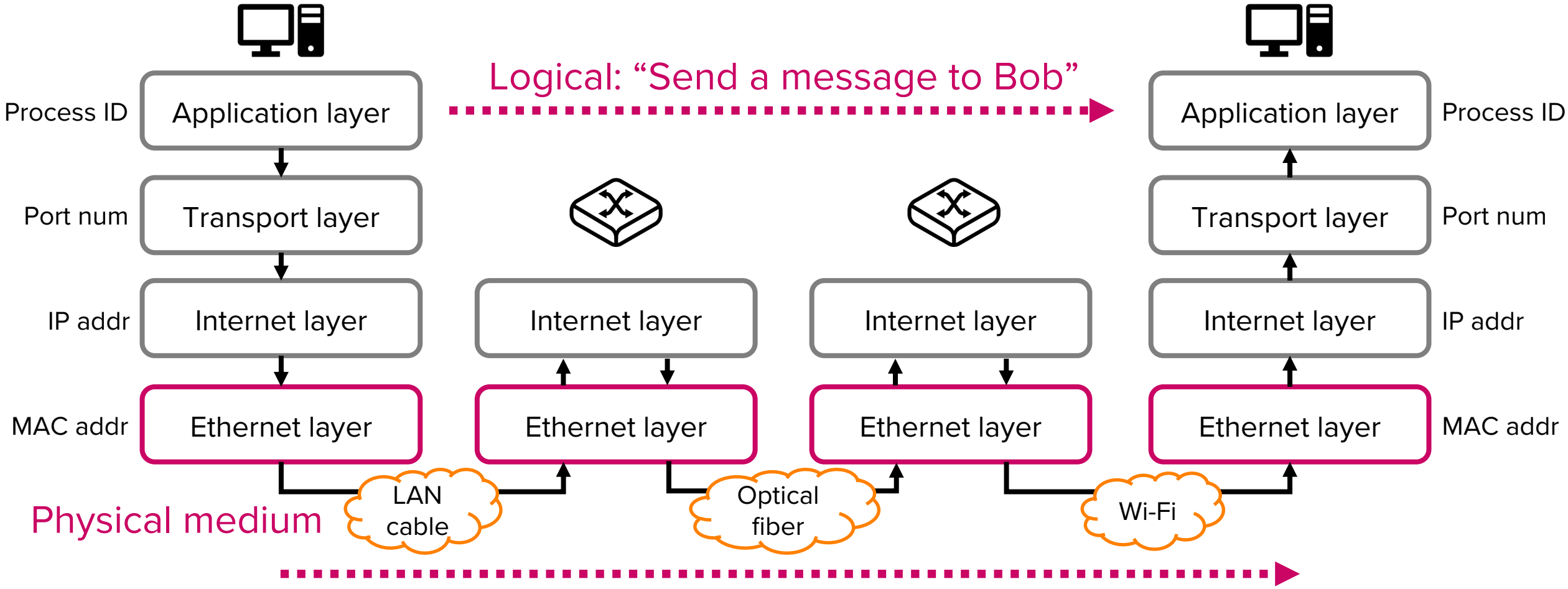
Background: Computer network

- Logical and physical data flow

Background: Computer network

POSTECH

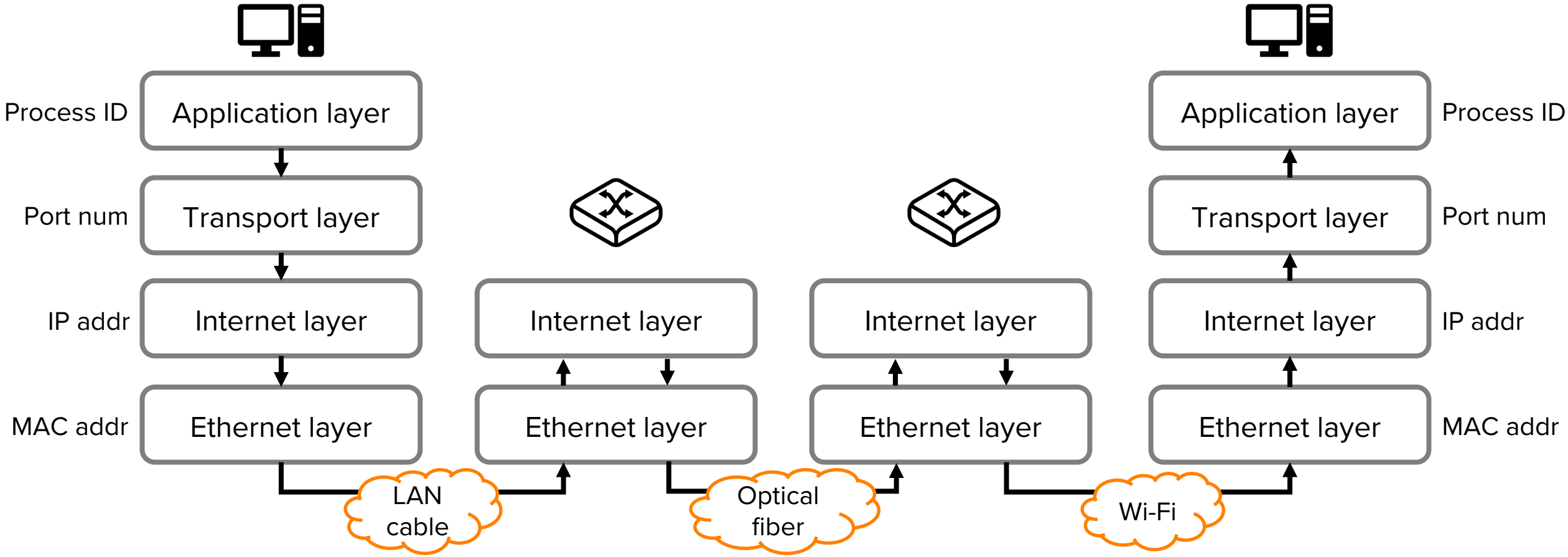
- Logical and physical data flow



Security?

POSTECH

Confidentiality, integrity, authenticity?

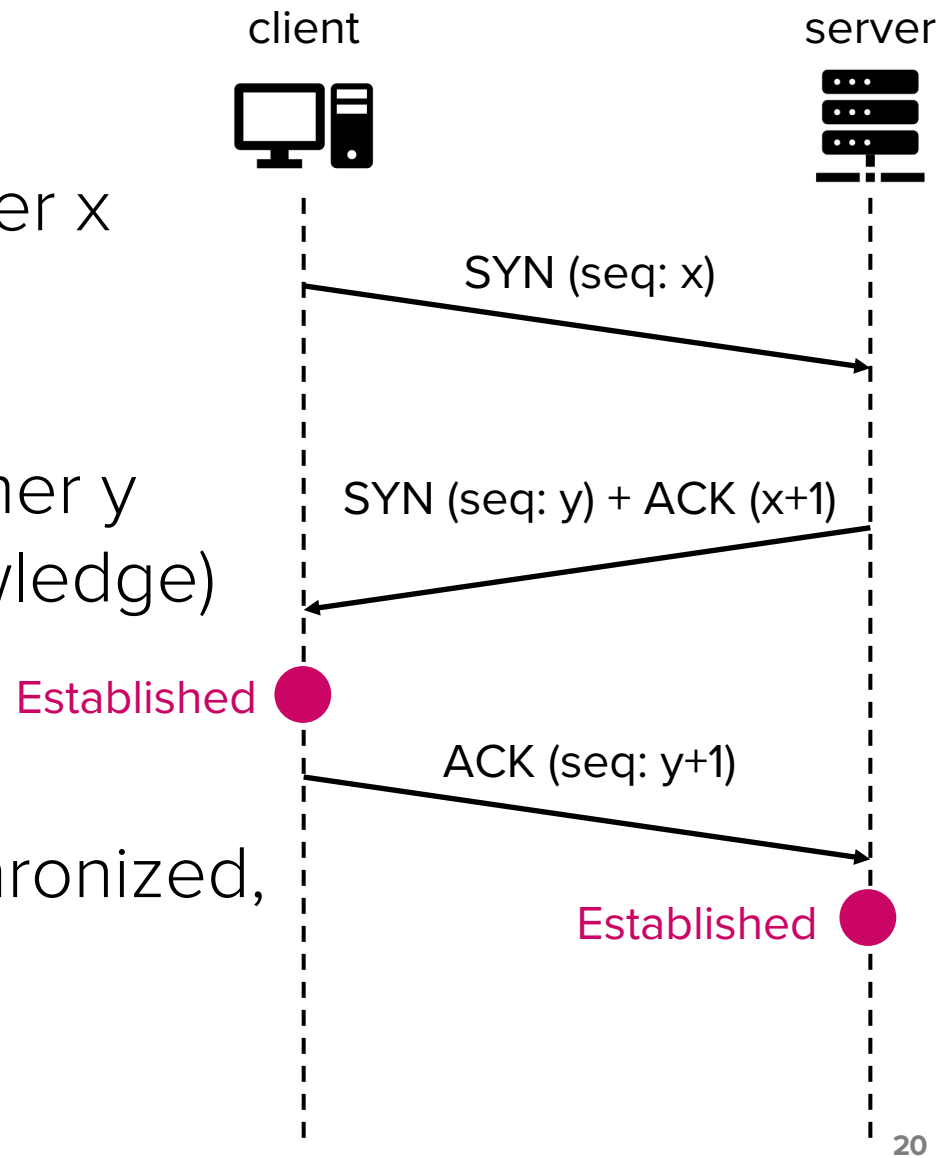


Background: Transport layer protocols

- TCP and UDP
  - Transmission Control Protocol (TCP): For reliably sending packets
    - 3-way handshake
      - Client SYN → Server SYN-ACK → Client ACK
    - Provides reliability (retransmissions) and ordering (sequence numbers)
  - User Datagram Protocol (UDP): For quickly sending packets without reliability
    - Connection-less
    - Does not provide reliability nor ordering

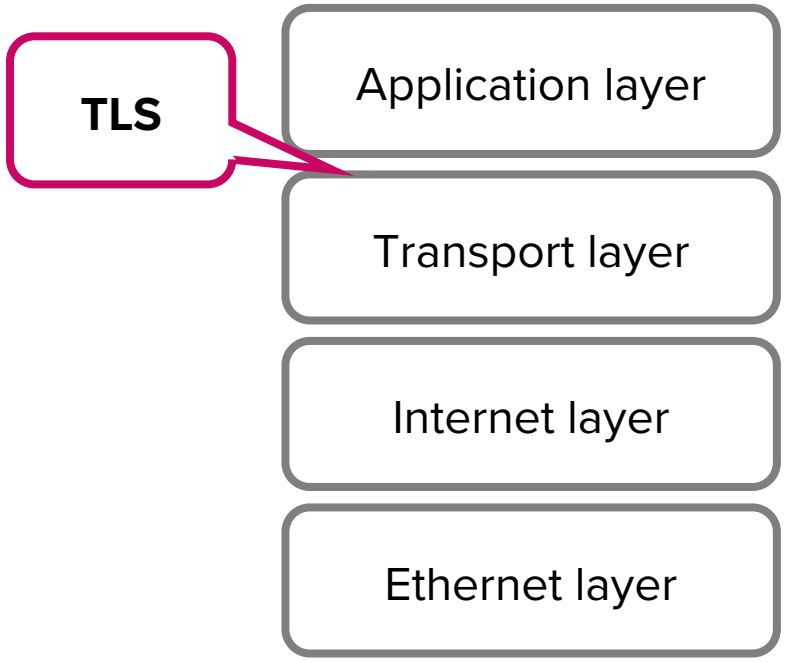
Background: TCP handshake

- 3-way handshake
  1. Client selects an initial sequence number x and sends a SYN (synchronize) packet to the server
  2. Server selects an initial sequence nubmer y and responds with a SYN+ACK (acknowledge) packet
  3. Client responds with an ACK packet
  4. Once the sequence numbers are synchronized, connection is established

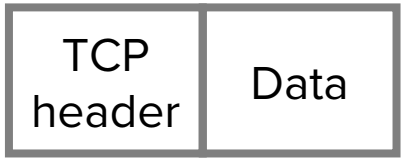


SSL/TLS protocol

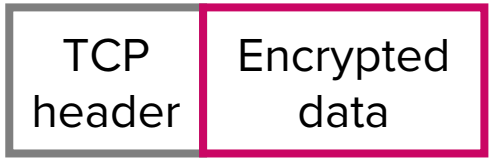
- Secure Sockets Layers protocol (SSL)
  - Outdated and replaced by TLS. “SSL” refers to TLS now
- Transport Layer Security protocol (TLS)
  - De facto standard for internet security
  - Goal: provide privacy and data integrity between two communicating applications
  - Built on top of TCP



TCP/IP network model



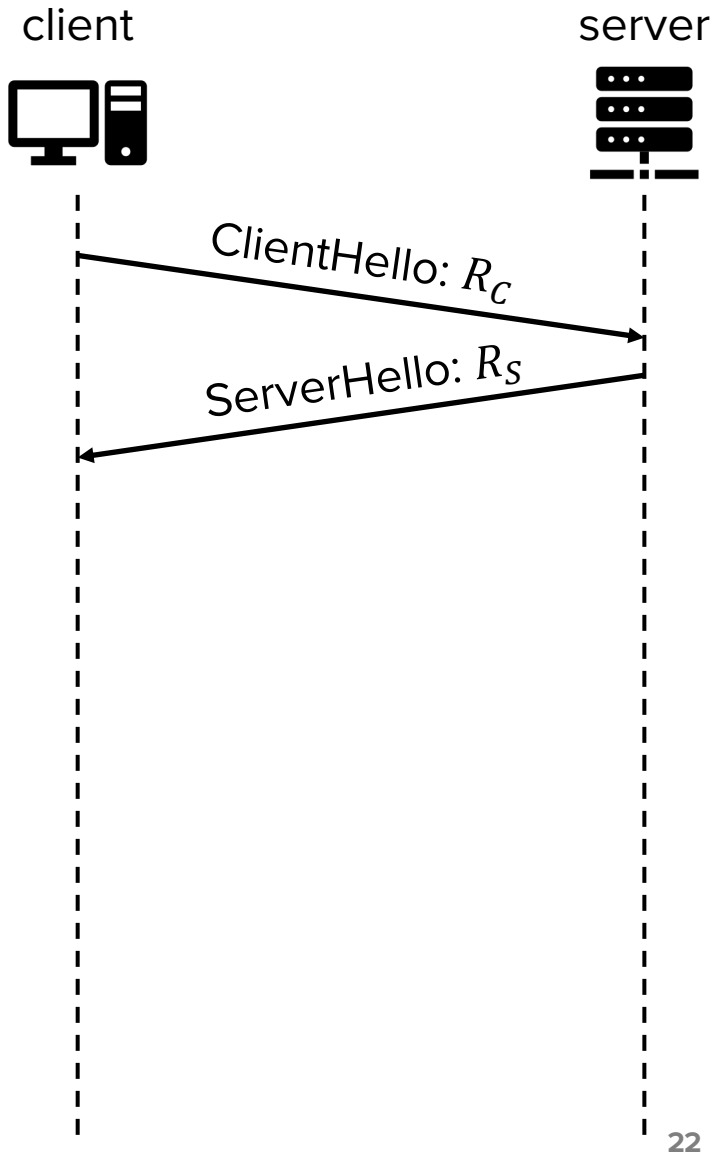
Transport layer encapsulation



Transport layer encapsulation with TLS

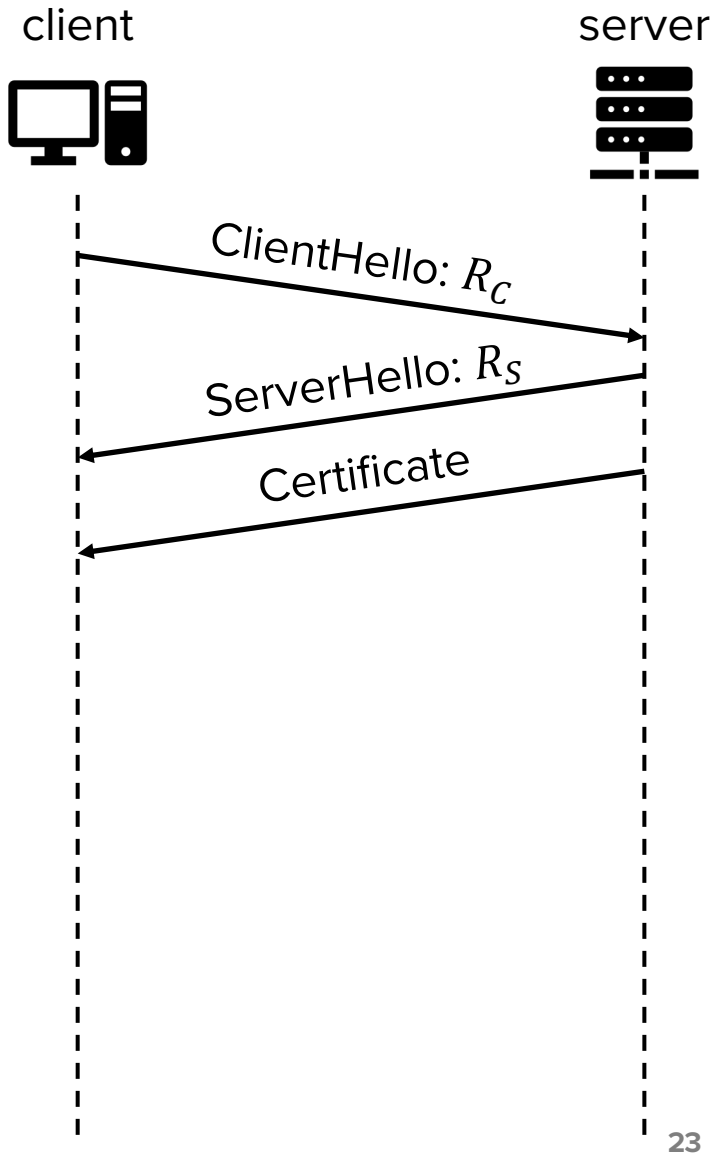
TLS handshake

- Step 1: Exchange hellos
  - Assumption: underlying TCP connection has already been established
  - Client sends **ClientHello**
    - A 256-bit random number R_C
    - A list of supported cryptographic algorithms
  - Server sends **ServerHello**
    - A 256-bit random number R_S
    - The algorithm to use (chosen from the client's list)
  - R_c and R_s prevent replay attacks



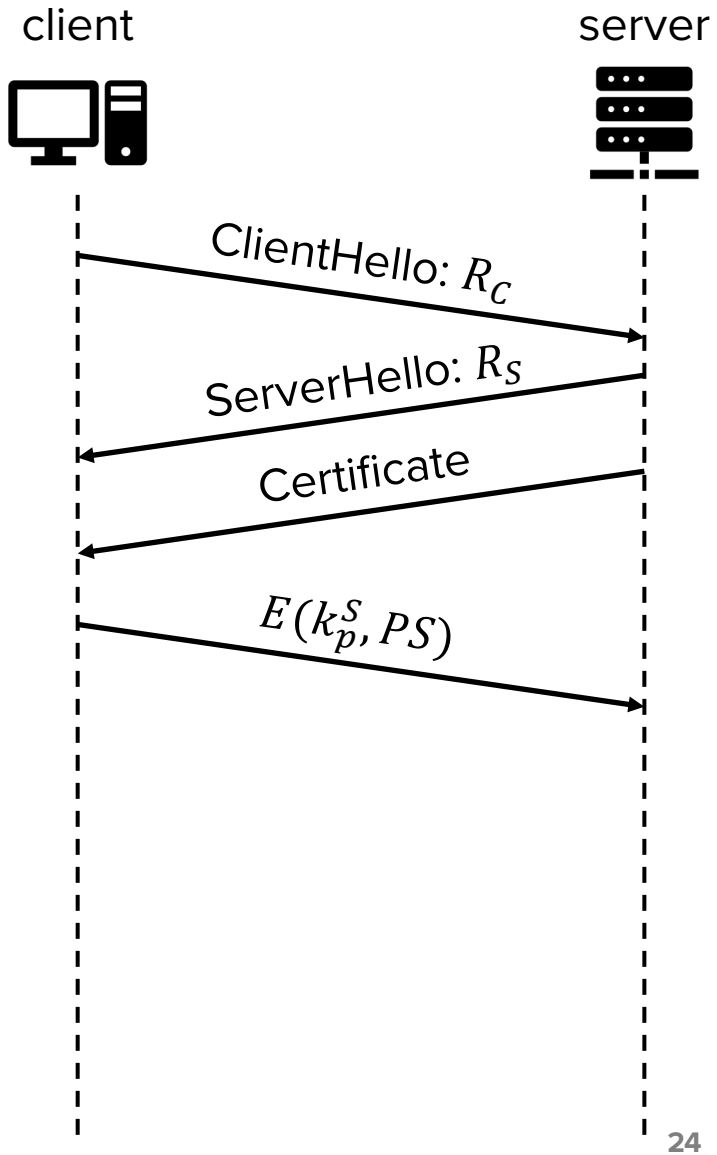
TLS handshake

- Step 2: Server sends its certificate
  - Recall: Certificate contains the server's identity and public key, signed by a trusted CA
  - Client validates the certificate
    - Verifies the signature
  - The client now knows the server's public key
    - Server's public key: k_p^S
    - The client is not yet sure if it is talking to the legitimate server (not an impersonator)
      - Certificates are public.
        Anyone can present anyone else's certificate



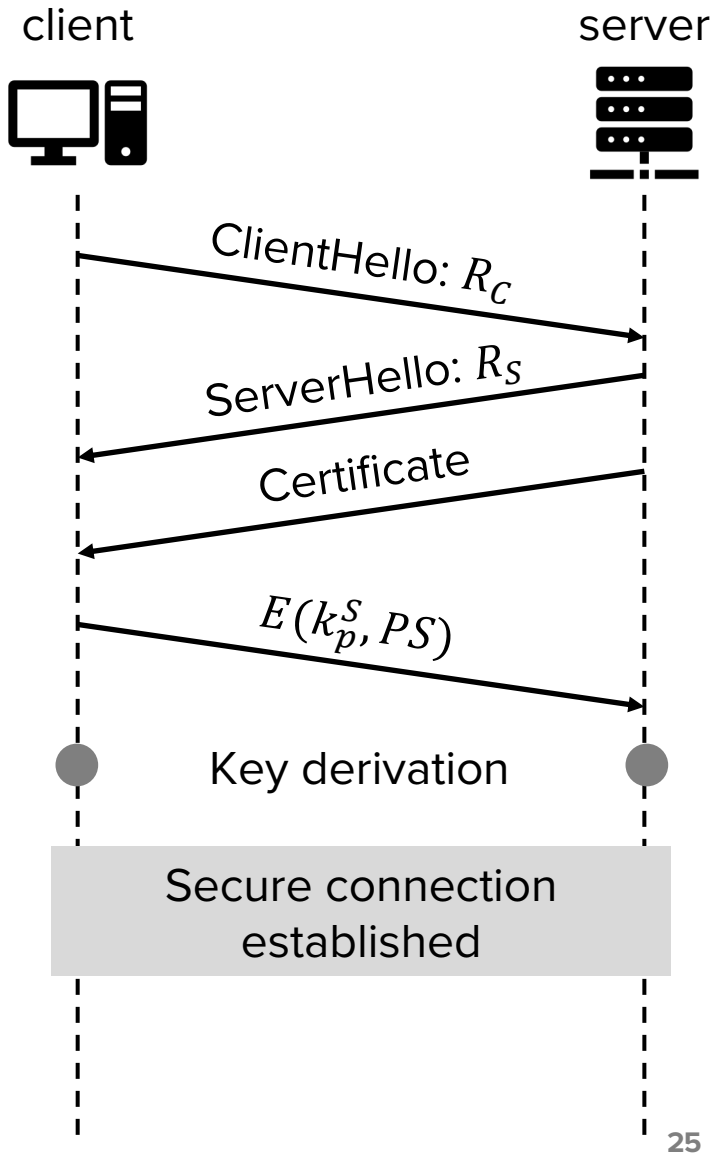
TLS handshake

- Step 3: Share premaster secret
  - The client randomly generates a premaster secret (PS)
  - The client encrypts PS with the server's public key and sends it to the server
  - The server decrypts PS using its secret key
    - If the server presents a valid PS, the client knows that the server is not an impersonator



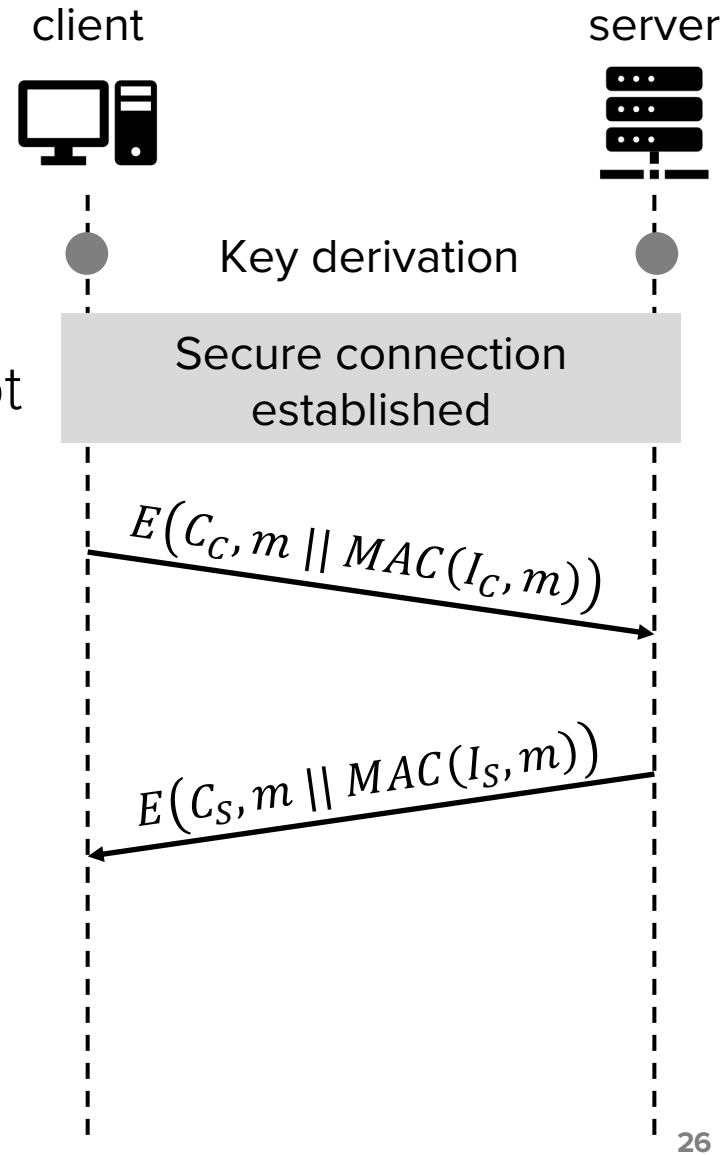
TLS handshake

- Step 4: Derive symmetric session keys
  - The server and client each derives session keys from R_C, R_S, and PS
    - Usually derived by seeding a PRNG with the three values
    - Any change results in different session keys
  - Four symmetric keys are derived
    - C_C, C_S: For encrypting c → s and s → c messages
    - I_C: For generating MAC of c → s messages
    - I_S: For generating MAC of s → c messages



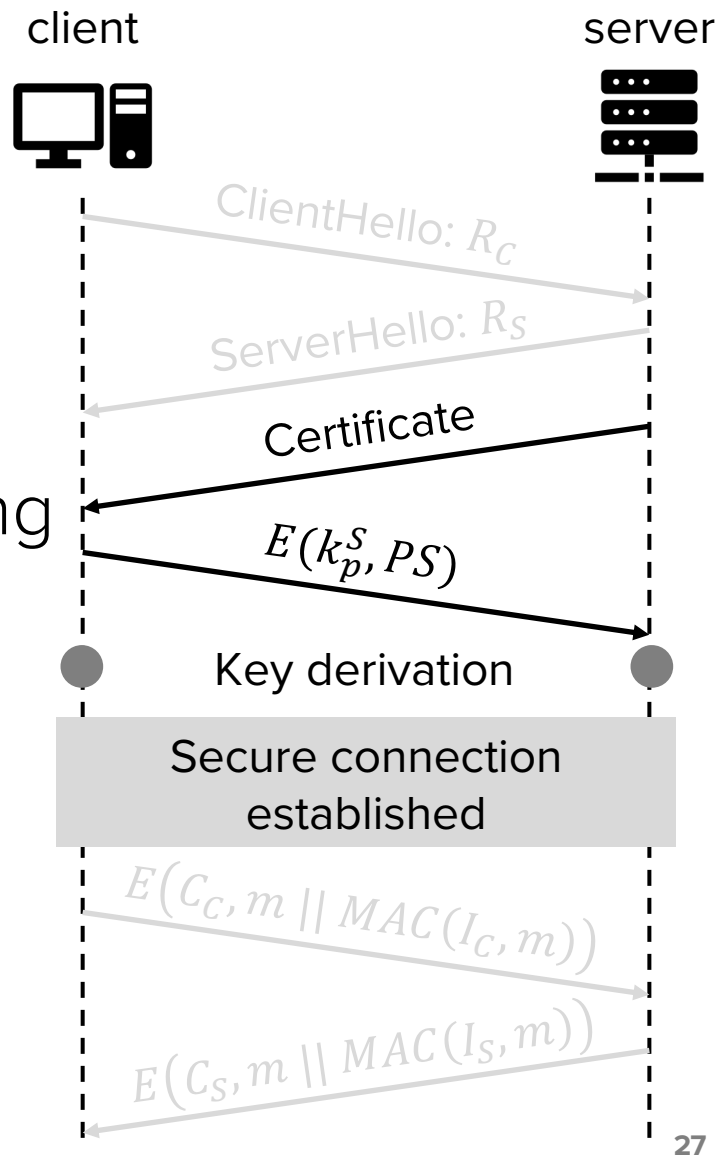
TLS message exchange

- Messages can now be sent securely
  - Utilize Authenticated Encryption (AE)
    - Using the derived session keys
    - Generate MAC of m, append MAC to m, then encrypt
    - Note:
      Even though Encrypt-then-MAC is considered safer (ref: Lec 11), TLS uses MAC-then-Encrypt for legacy reasons



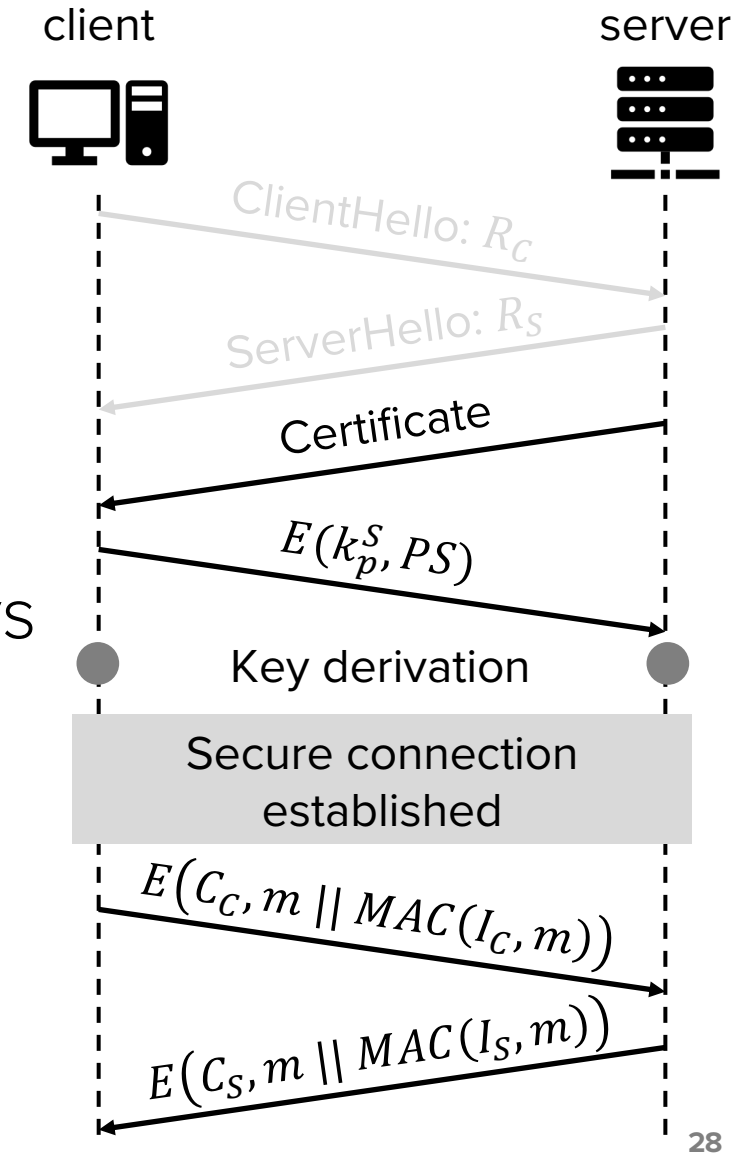
Security of TLS

- Authenticity: Can we make sure we are talking to the legitimate server?
  - The server sent its certificate, so we know its public key k_p^S
  - The server proved that it owns the corresponding secret key k_s^S
    - It decrypted the premaster secret PS
  - An impersonator cannot derive the same session keys as he/she does not own the secret key to decrypt the PS



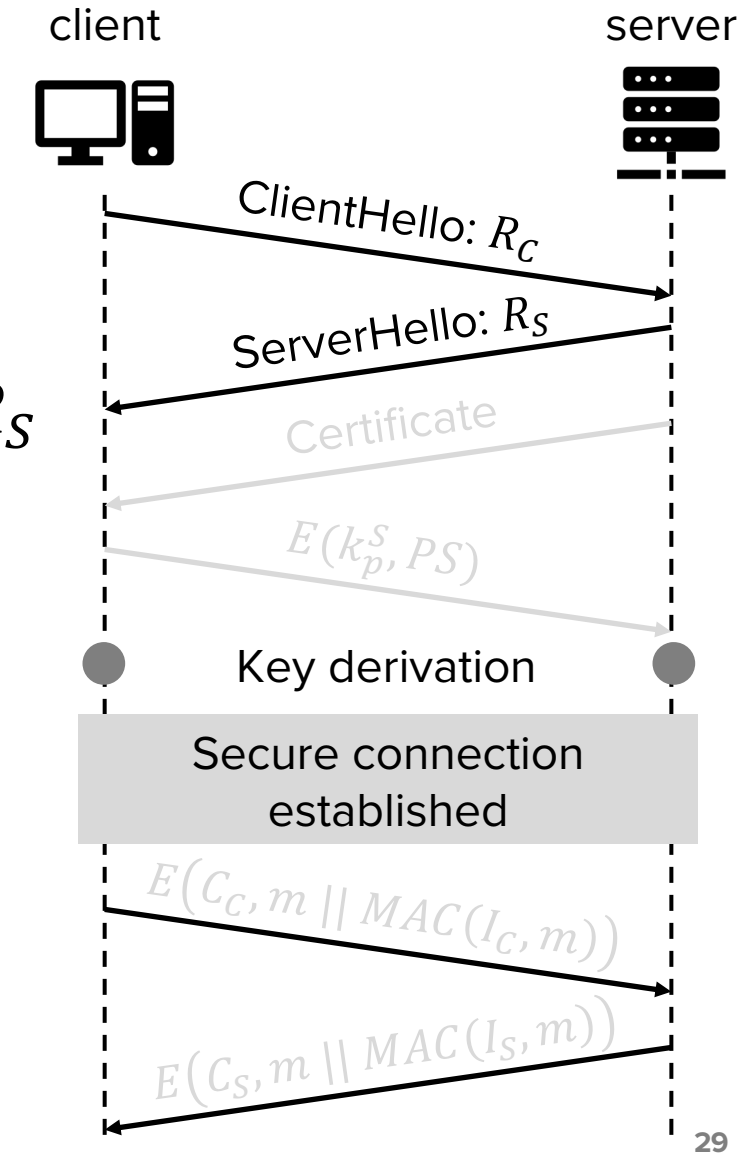
Security of TLS

- Confidentiality and Integrity: How can we ensure that attackers cannot read or tamper with our messages?
  - The attacker does not know PS
  - The session keys are derived from PS
  - Authenticated encryption using the session keys provide confidentiality and integrity



Security of TLS

- Replay attacks: How can we ensure that an attacker is not replaying old messages from a past TLS connection?
  - Every TLS handshake uses a different R_C and R_S (random numbers)
  - The session keys are derived from R_C and R_S
    - These keys are different for every TLS connection



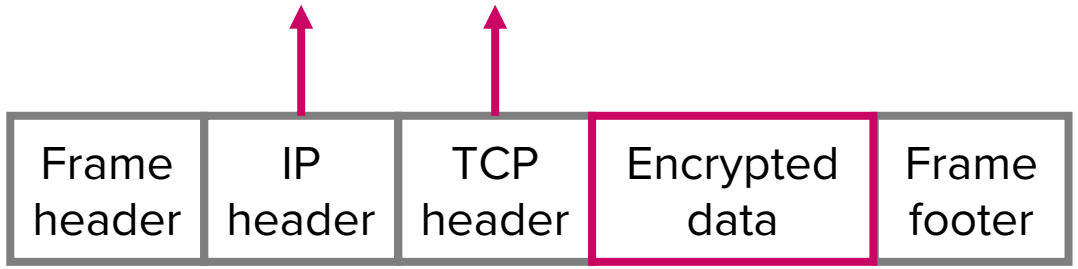
What TLS does and doesn't

- TLS guarantees end-to-end security
  - Even if every entity between the client and the server is malicious, TLS provides a secure communication channel
  - Examples
    - A local attacker captures all Wi-Fi communications
      - The attacker cannot read TLS messages w/o session keys
    - A MitM tries to inject TCP packets
      - These Packets will be rejected (cannot generate valid MACs w/o session keys)
  - Note, end-to-end security does not help if one end is malicious (e.g., communicating with a malicious server)

What TLS does and doesn't

- TLS does not guarantee anonymity
  - Anonymity: Hiding the client's and server's identities from attackers
  - Attackers can figure out who is communicating with TLS
    - Server's certificate, containing server's identity, is sent during the handshake
    - Attacker can see IP addresses and ports of underlying IP and TCP protocols

Required for locating src/destination (cannot be encrypted)



Encapsulation after TLS

What TLS does and doesn't

- TLS does not guarantee availability
  - Availability: Keeping the connection open in the face of attackers
  - Attackers can hijack and drop TLS packets to stop TLS connections
  - In other words, TLS connections can still be censored
    - South Korean government blocks access to porn and gambling websites

TLS Application

- Secure e-commerce using TLS (e.g., online shopping)
  - Confidentiality and integrity matters
    - Do not need anonymity
  - Client authentication is not needed until the client decides to buy something
  - TLS provides secure channel for sending credit card information
  - Client authenticated using credit card information, merchant bears (most of) risk

HTTPS

HTTPS: HTTP over SSL

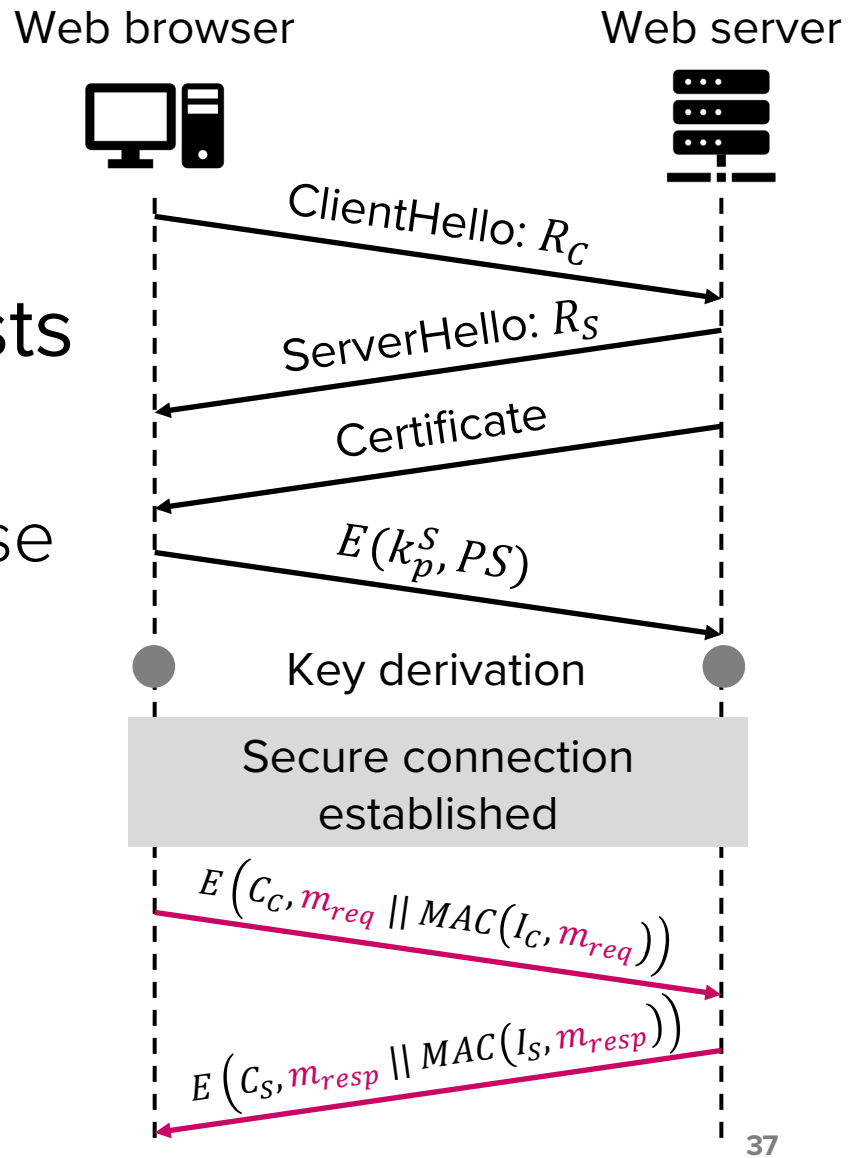
- Combination of HTTP (Hypertext Transfer Protocol) and SSL
  - TLS applied specifically for securing the communication between a web browser and a web server
- All modern browsers support HTTPS protocol
  - If not, avoid at all costs!
- URLs of the servers providing HTTPS connection start with https://

HTTPS: HTTP over SSL

- Connection
  - HTTP connection uses port 80
  - HTTPS connection uses port 443, which invokes TLS protocol
- Encryption
  - URL of the requested document (web page)
  - Contents of the document
  - Contents of browser forms (e.g., username and password)
  - Cookies between the server and the browser
  - Contents of HTTP header

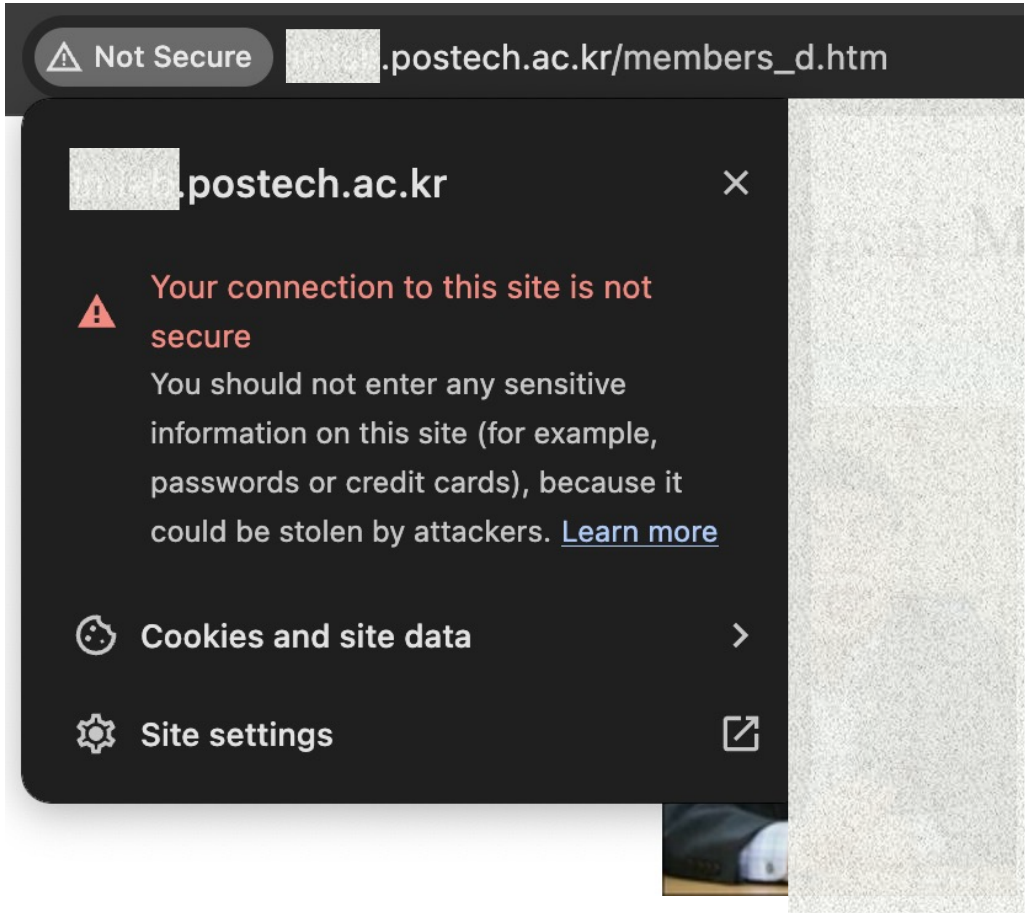
HTTPS connection

- The web browser initiates a TLS connection to the web server
- After the TLS handshake, all HTTP requests and responses are encrypted
  - i.e., m_{req}: HTTP request, m_{resp}: HTTP response

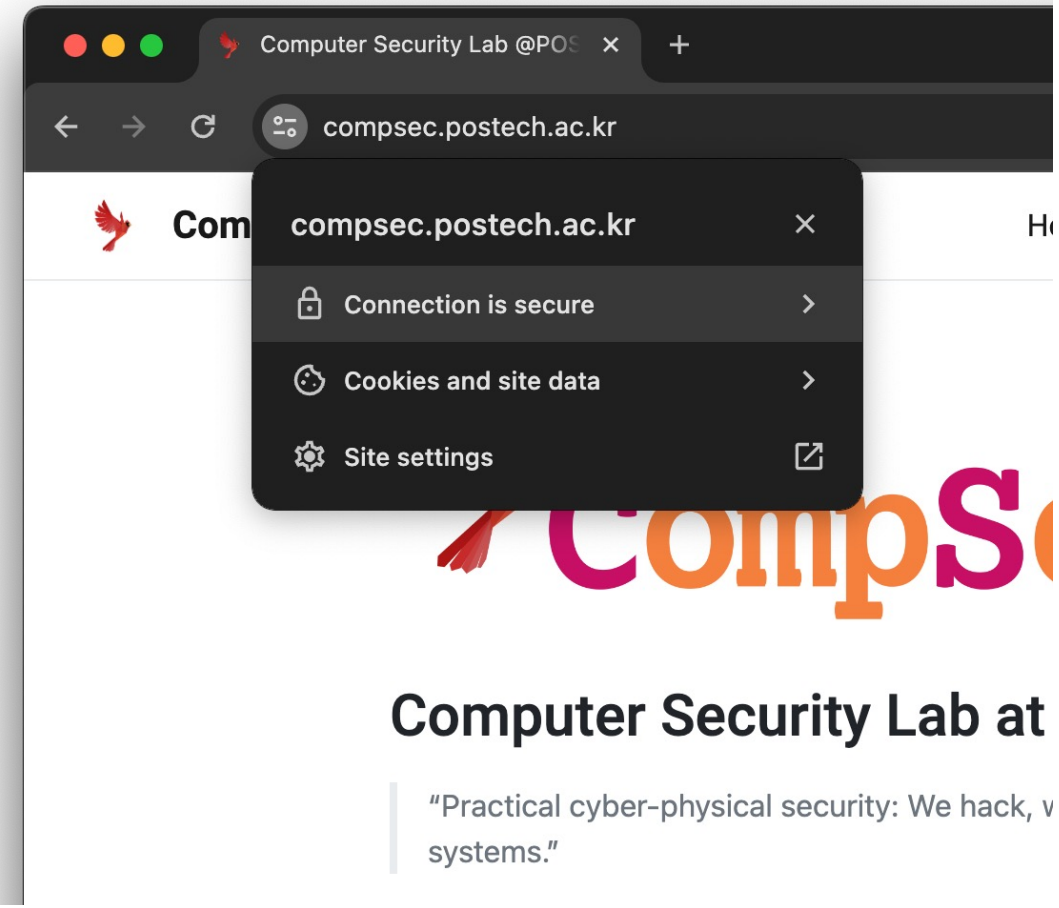


HTTP vs HTTPS

http://[redacted].postech.ac.kr

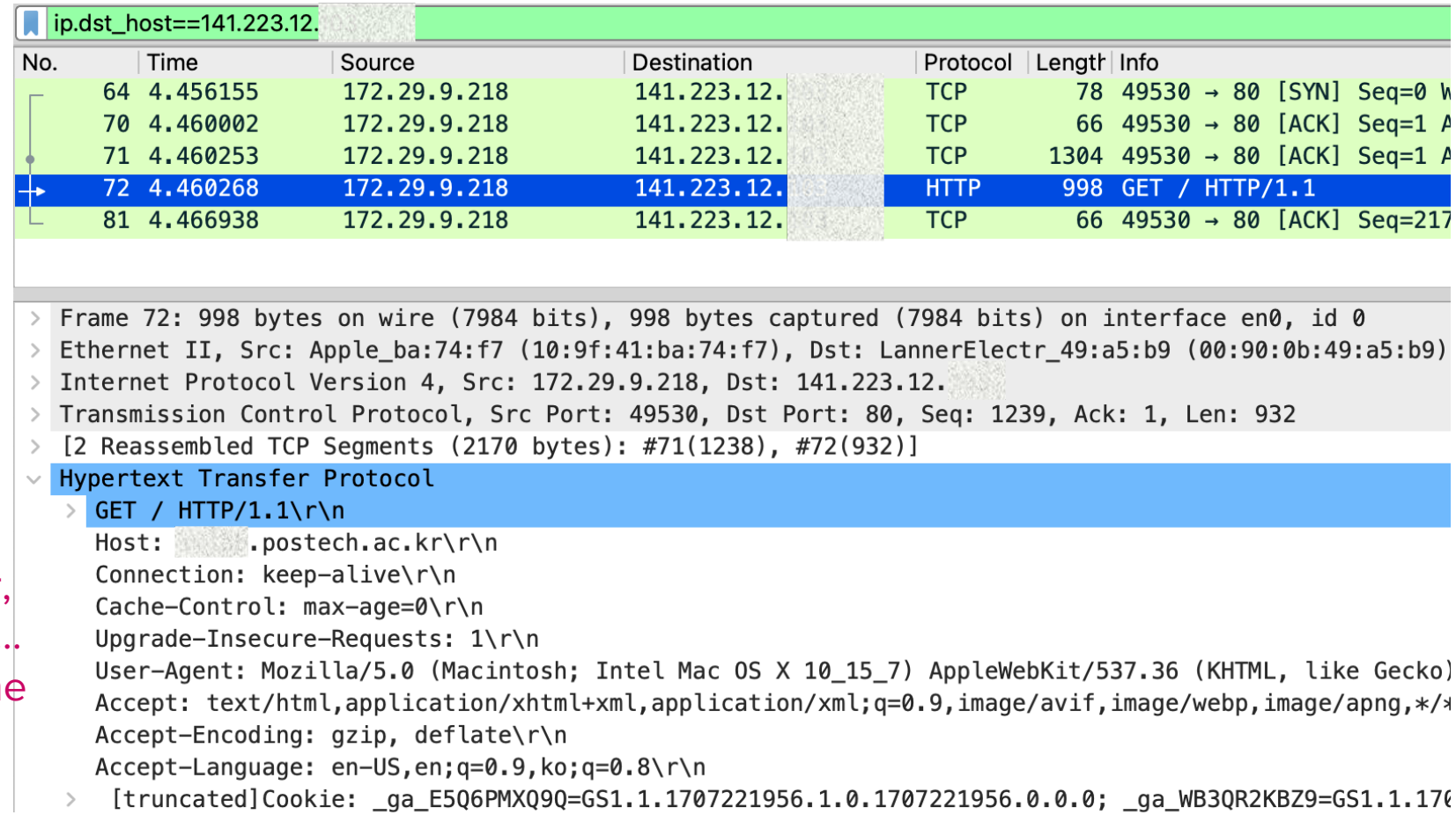


https://compsec.postech.ac.kr



HTTP vs HTTPS

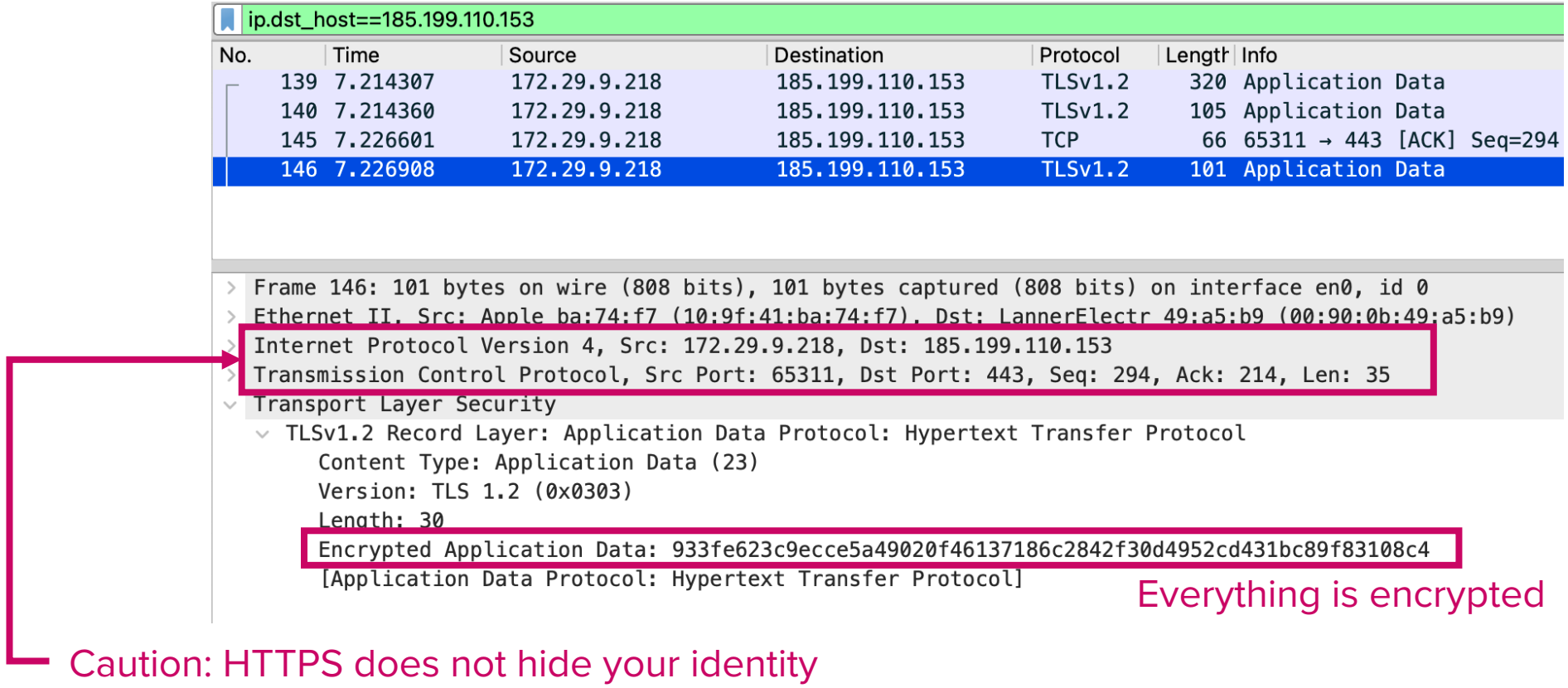
- HTTP packet captured through Wireshark



URL, HTTP header, contents, cookies, … are visible to anyone

HTTP vs HTTPS

- HTTPS packet captured through Wireshark



Remaining Issue: Can we trust CAs?

Recall: Certificates in TLS

- The server sends its certificate
  - Server's identity (domain name) and its public key signed by CA's secret key
- The browser verifies the server's certificate
  - CA's public key is embedded in the browser (ref: lec 12, trust anchor)
  - The browser uses the CA's public key to verify the certificate
- Once verified, the browser trusts the server's public key

Issues: Unknown CA

- What if the browser does not have the CA's public key?
  - Not all CA information is embedded
  - Typical behavior: Warn the user that the website is not verified
    - Connection can still be established, but the server's legitimacy is not assured

- Potential problem
  - The server indeed is a malicious server
  - End-to-end security is broken

Issues: Revocation

- What if an attacker steals a server's private key?
  - The certificate with the corresponding public key is no longer valid
  - TLS certificates have an expiry date, but it takes time to expire
- Solution: Certificate revocation lists (CRL) (ref: Lec 12)
  - The CA occasionally sends out lists of certificates that should be revoked
  - The browser occasionally downloads the lists

Issues: Trust anchors

- Recall: We designate multiple trust anchors to solve the single-point-of-failure problem
- A CA might issue malicious/fraudulent certificate
  - A CA gets hacked
  - An attacker bribes the CA to issue a fraudulent certificate
- Problem: Too many trust anchors
  - Modern browsers trust 100~200 CAs
  - One compromised CA is enough for attackers to launch large-scale attacks (the weakest link matters!)

Issues: Trust anchors

- Real-world incidents: Comodo SSL certificate breach (2011)
  - Comodo was a major CA
    - Not anymore
  - Comodo's account was compromised
  - Iranian hacker issues nine fraudulent SSL certificates for popular websites, including Gmail, Hotmail, Skype, …
  - (although not known) the hacker could impersonate these websites and intercept all TLS-encrypted traffic
    - Again, end-to-end security is broken if one end is malicious

Issues: Trust anchors

- Real-world incidents: DigiNotar server hack (2011)
  - DigiNotar was another major CA
  - Attacker (alledgedly backed by Iranian government) compromised all eight certificate-issuing servers of DigiNotar
  - The attacker issued more than 500 fake certificates, including Google's

Issues: Trust anchors

- Real-world incidents: Symantec mis-issuance (2017)
  - One of the largest providers of TLS certificates (not anymore)
  - Symantec issued certificates without sufficiently validating the identity of the entities requesting them
    - They issued certificates for domains without verifying ownership
      - "Hey Symantec, I own google.com. Can you issue a certificate?" → "Absolutely"
    - They issued certificates for non-existent domains
      - "Hey Symantec, I own xcvbmnzgirsjcxv.com. Can you issue a certificate?" → "Absolutely"
  - 30K problematic certificates were issued, according to Google
  - Chrome and Firefox revoked all Symantec-issued certificates

Still an unsolved problem!

Modern CA example: Let's Encrypt

- To use TLS, every web server needs to obtain and maintain certificates
  - Most certificate providers charge money for issuing TLS certificates
- Let's Encrypt (LE) issues certificates for free
  - Web server requests a certificate
  - LE sends the server a file to be uploaded
  - The server uploads the file to the website
  - LE verifies that the file has appeared on the website
  - Identity verified (domain & ownership) → LE issues a certificate

Coming up next

POSTECH

- Review for midterm exam
  - Lab assignments
  - Topics covered in class
  - Q & A

Questions?