Week 13: Cryptography Applications (P1)

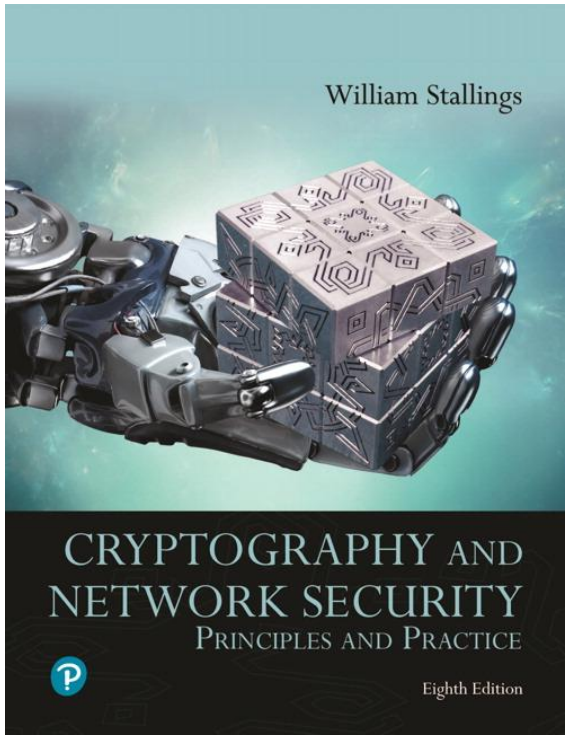
PhD. Ngoc-Tu Nguyen
tunn@uit.edu.vn

Outline

- Network secure protocols
  - Authentication;
  - Key agreement;
  - Cryptographic negotiation;
  - SSL/TLS; SSH; IPSec; Kerberos;
- Blockchain-base network
- Lattice-bases cryptography and Post-quantum security

Textbooks and References

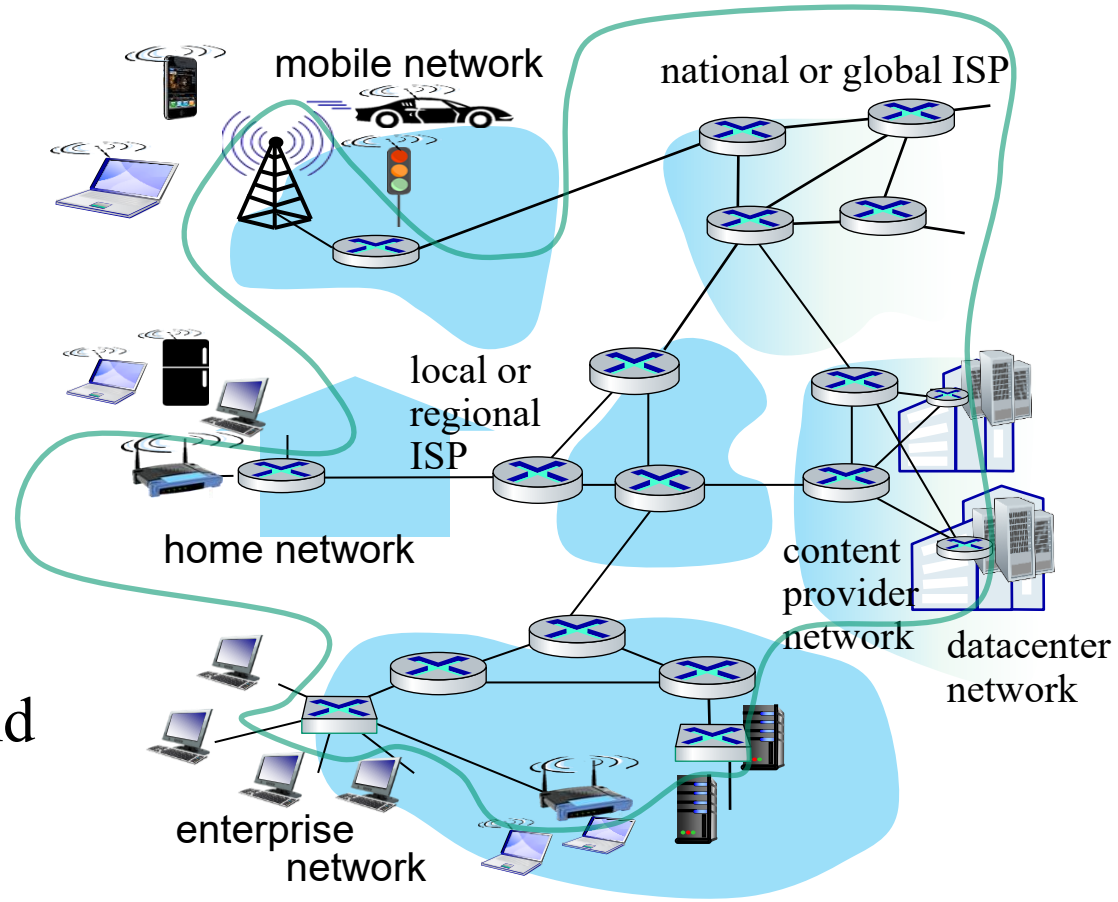
- Text books



[1] Chapter 14.15

Motivations

- **Network devices:**
  - End devices
  - Intermediate devices
- **Communication links**
  - fiber, copper, radio
  - transmission rate: bandwidth
- **Network protocols**
  - How to secure storage and exchange data?
    - Send?
    - Receiver?
    - Storage?



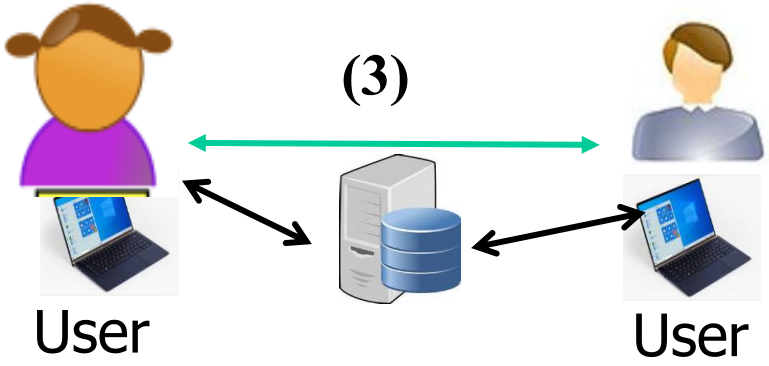
Network secure protocols (P1)

- Authentication
  - Public key (Certificate) approach
  - Prior secret-shared approach
- Cryptographic algorithm negotiation
  - Ciphers, MAC
- Key agreement
  - Diffie-Hellman key exchange + extra…
- Secure protocol implementation
  - Chosent layer to implement
  - Chosen cipher to encrypt exchange data

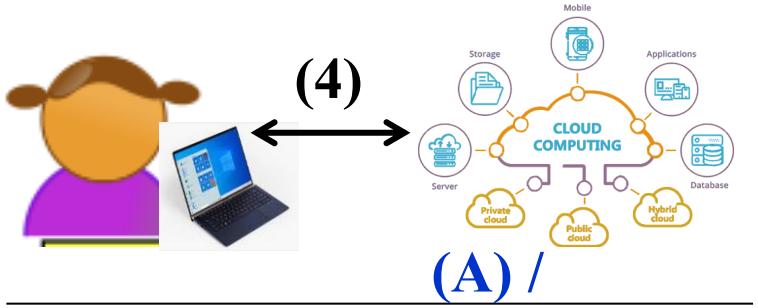
Motivations

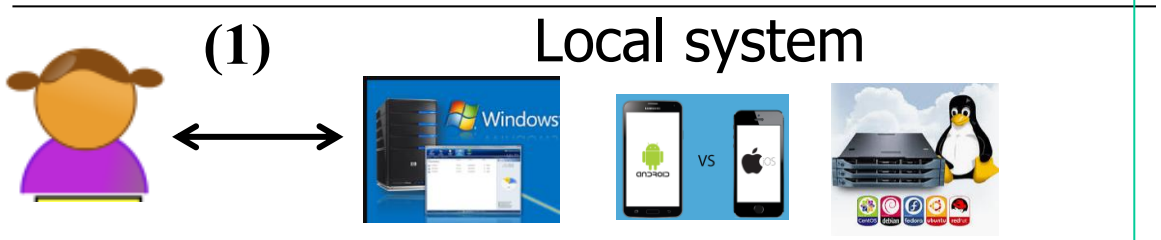
Secure protocool goals:

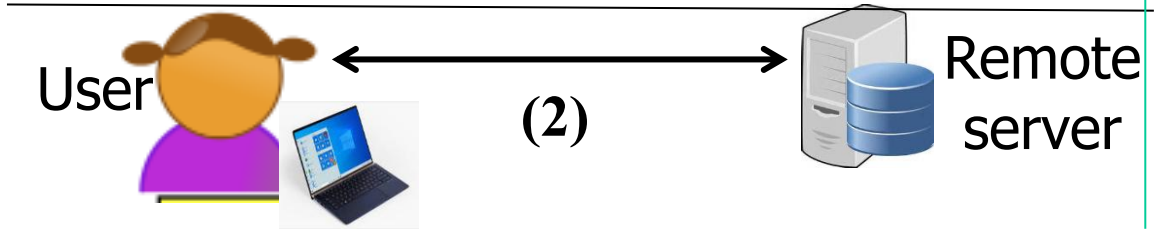
- ✓ **Mutual authentication**
- ✓ **System parameters and cryptographic algorithms?**
- ✓ **Key agreement**



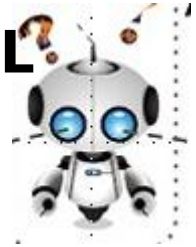
Iot/Edge/fog/cloud network







(5) **AI/ML**



Authentication and Authorization

❑ Authentication (users, hosts, process or programs)?

Authentication= "Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system"

Authentication process=
- Identification information
- Verification

📖 https://csrc.nist.gov/glossary/term/authenticationation

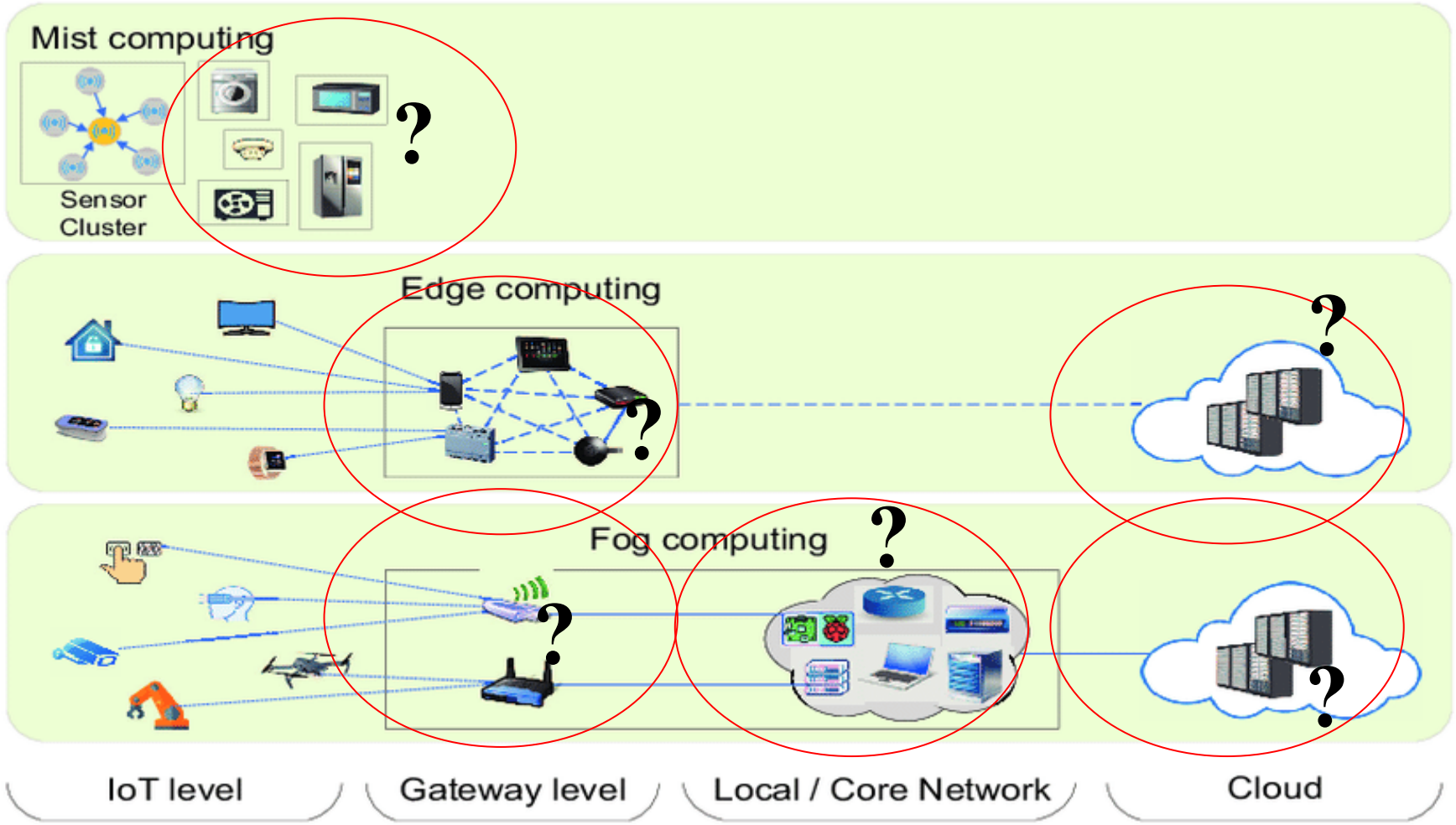
❑ Authorization

The right or a permission that is granted to a system entity to access a system resource.

📖 https://csrc.nist.gov/glossary/term/authorization

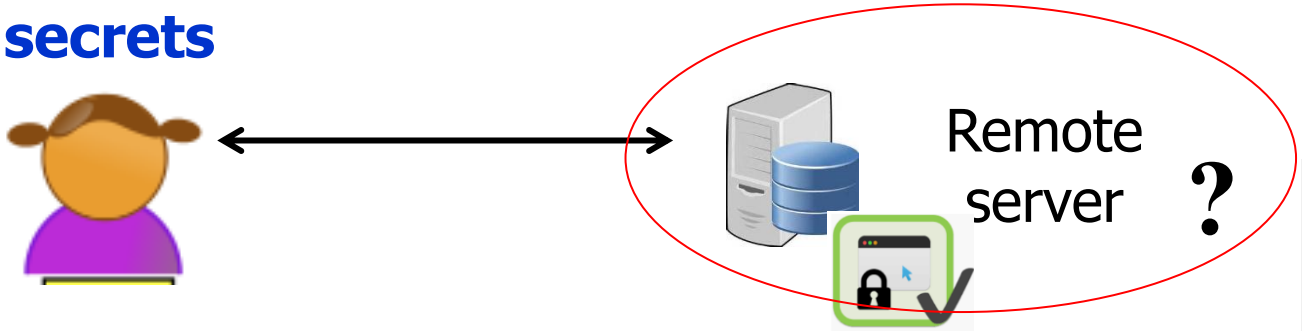
Authentication

- **Users authenticate server/host/ resources**

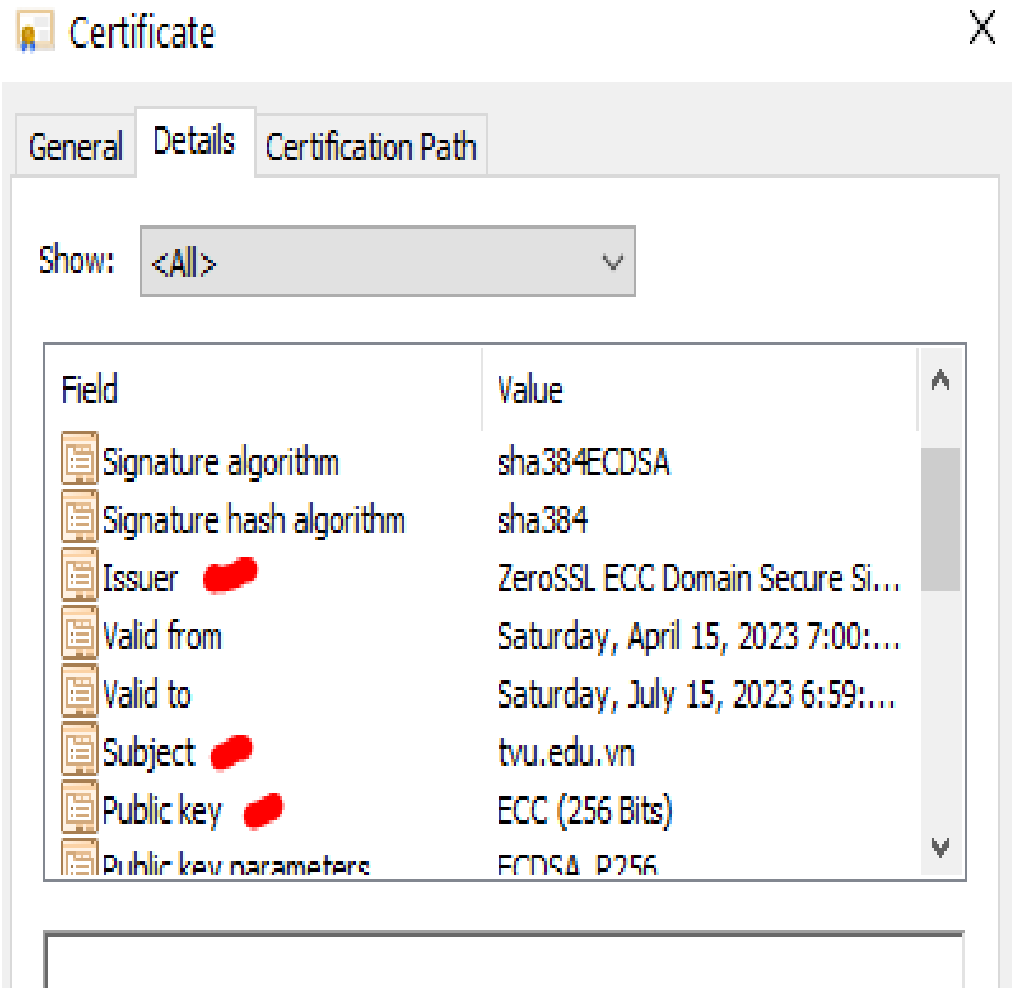


Authentication

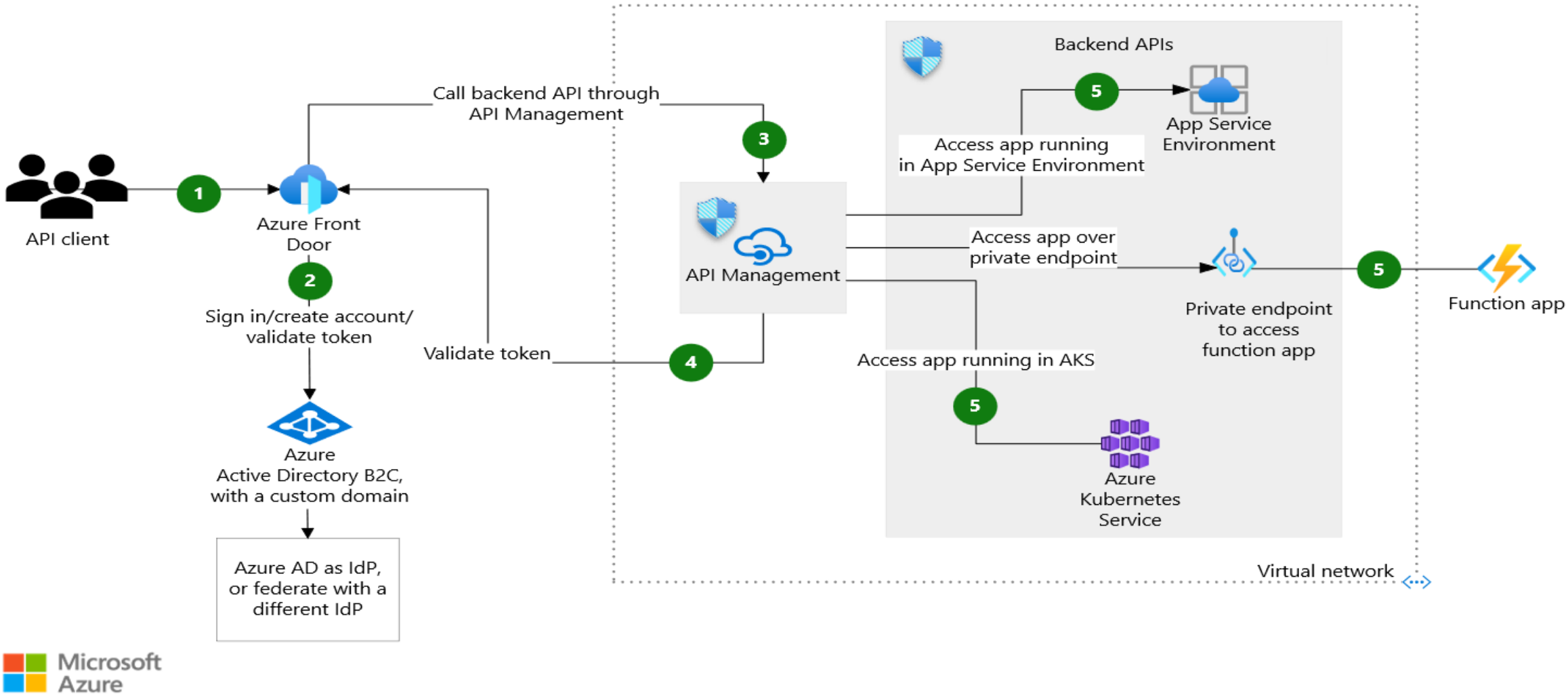
- **Users authenticate server/ nodes/ resources: digital certificate/pre-share secrets**



openssl s_client -connect www.facebook.com:443



Authentication and Authorization



Authentication

- **Authentication factors**

- Knowledge (something the user/node knows)

ID, Password, PIN, answers to prearranged questions

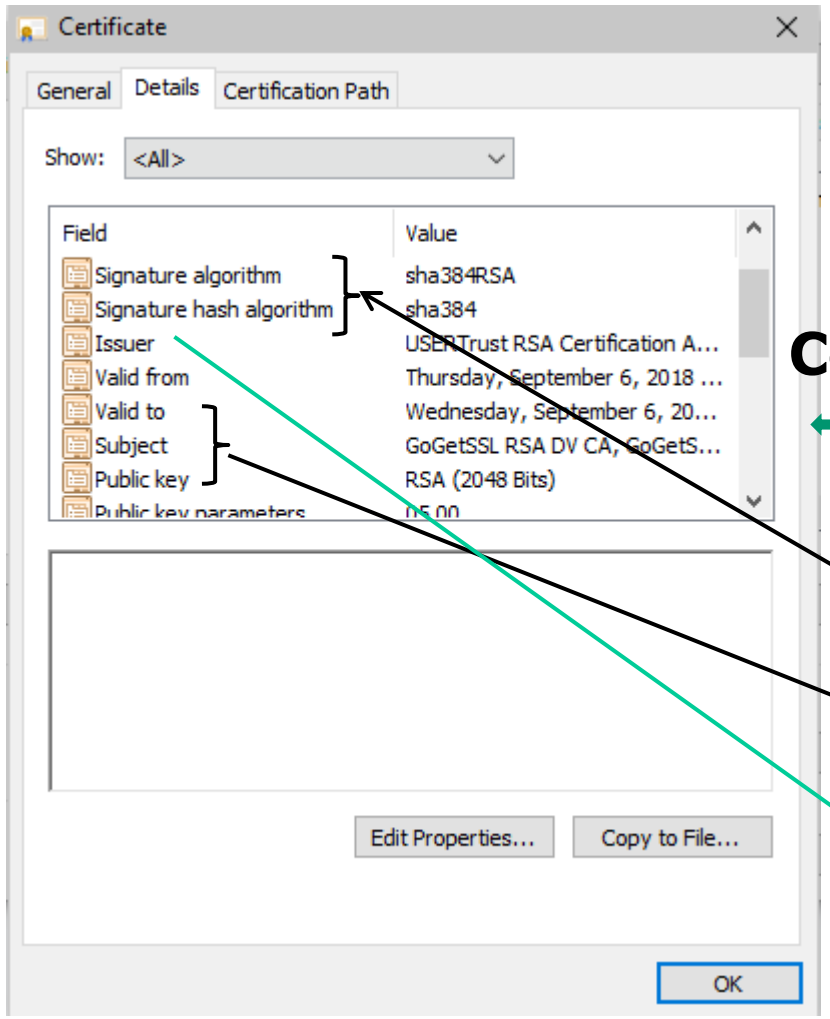
- Possession (something the user/node has)

Smartcard, electronic keycard, physical key, user's devices, digital certificates, …

- Inherence (some physical characteristic of the user/devices)

Fingerprint, retina, face, voice pattern, handwriting, typing rhythm, PUFs,…

Authenticate server/resources



Certificates-based authentication

Remote server

Certificate

Public key/ Secret key

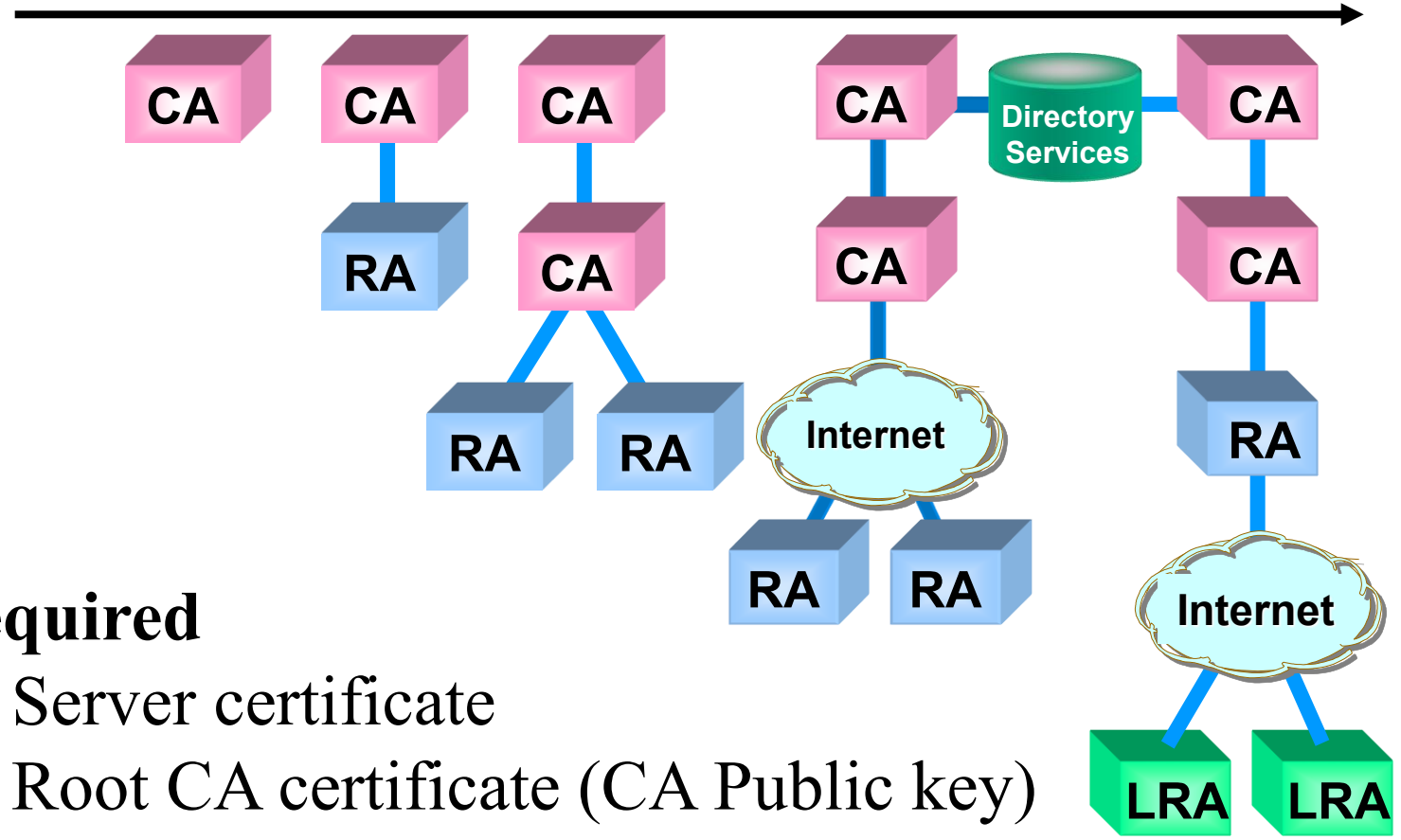
- Exchange authentication/
- key agreement data

Combine:

- Public key
- Server Identity (IP, name,...)

Trusted CA **PKI**

Certificate implementation



Required

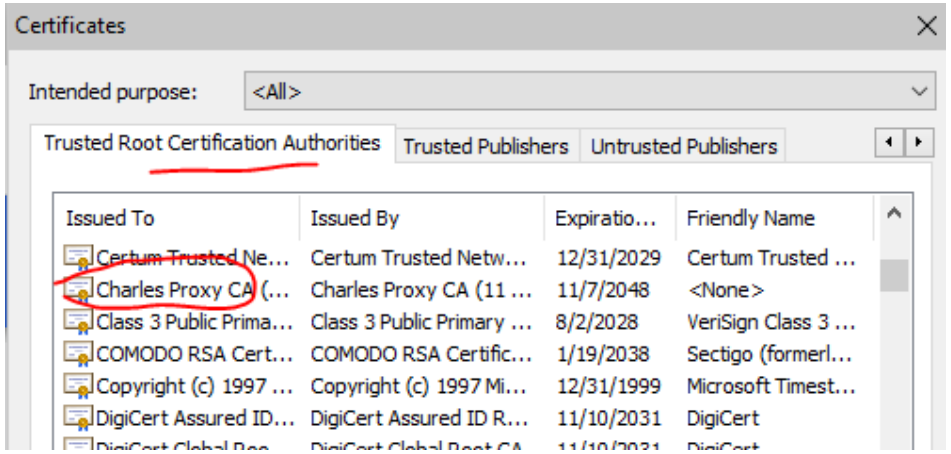
- Server certificate
- Root CA certificate (CA Public key)

openssl s_client -connect facebook.com:443 -showcerts

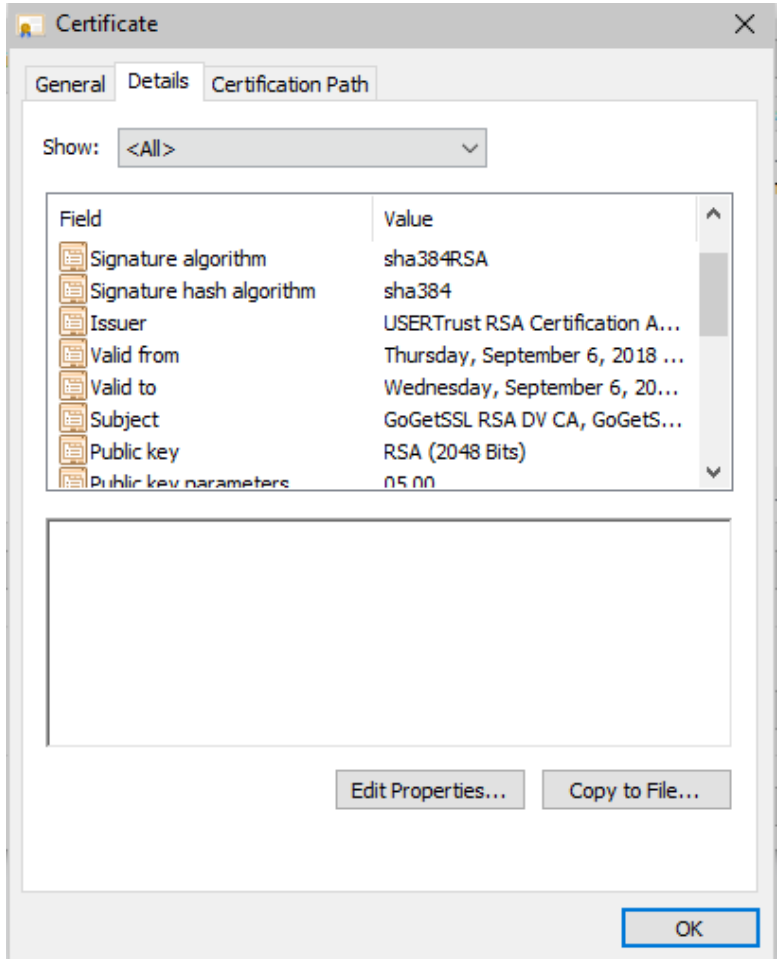
Authenticate server/resources

❑ Limitation

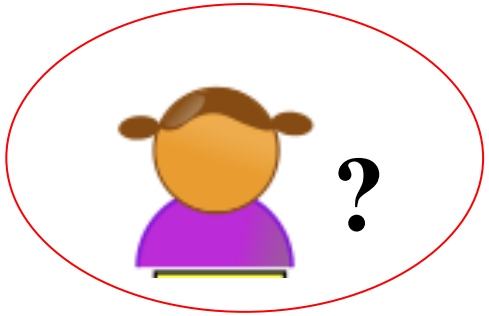
- **Need PKI**
- **Revocation?**
- **Network Payload?**
- **Trusted CA?**
  - ✓ **Semi-Trusted**
  - ✓ **Zero-Trusted**



Certificates-based authentication



Authenticate user/end-devices



Remote server

Server Certificate

(SK_S, PK_S)

$verify\ (certificate, PK_{root}) \Rightarrow PK_S$

- How the server authenticate the user?

sk_{shared}

ex. pw

Using pre-shared secrets

$store\ sk'_{shared}$

ex. $h(pw)$

Authenticate user/end-devices

(1) Password-based Authentication

$store: name,\ h(pw)$

$name, pw$

- **Login**

	UserName	HashedSaltedPwd	UserID
1	A-JayBibbins637	0x600FAD66D16A56AF3459D9C996DFF98B	10000
2	A-JayTorain976	0x4E30184141CB6CFF4120D6AC443CCE54	10001
3	AadamDobbin507	0xED2A51BD92E1EE8333314407817CD6F9	10002

- Provide $name, pw'$ to server?
  - Locate the row using name
  - Check $h(pw')? = h(pw)$

  ⟹ authenticate the user

Authenticate user/end-devices

Password-based Vulnerabilities

Pre-shared secrets

Human-memorable password

Small space of possible passwords

(1) Dictionary attacks

(2) Verification table attacks

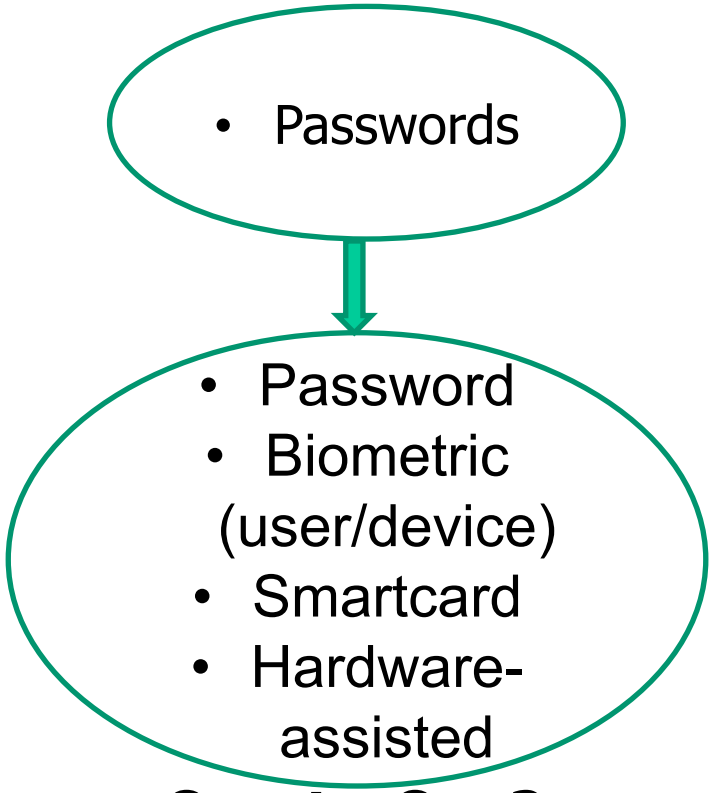
	UserName	HashedSaltedPwd	UserID
1	A-JayBibbins637	0x600FAD66D16A56AF3459D9C996DFF98B	10000
2	A-JayTorain976	0x4E30184141CB6CFF4120D6AC443CCE54	10001
3	AadamDobbin507	0xED2A51BD92E1EE8333314407817CD6F9	10002

Offline Dictionary attacks

https://hashkiller.io/listmanager

Authenticate user/end-devices

(2) Multiple factor Authentication



- **One-to-One?**
- **One-to-many?**

Scale-up ability?

Pre-shared secrets

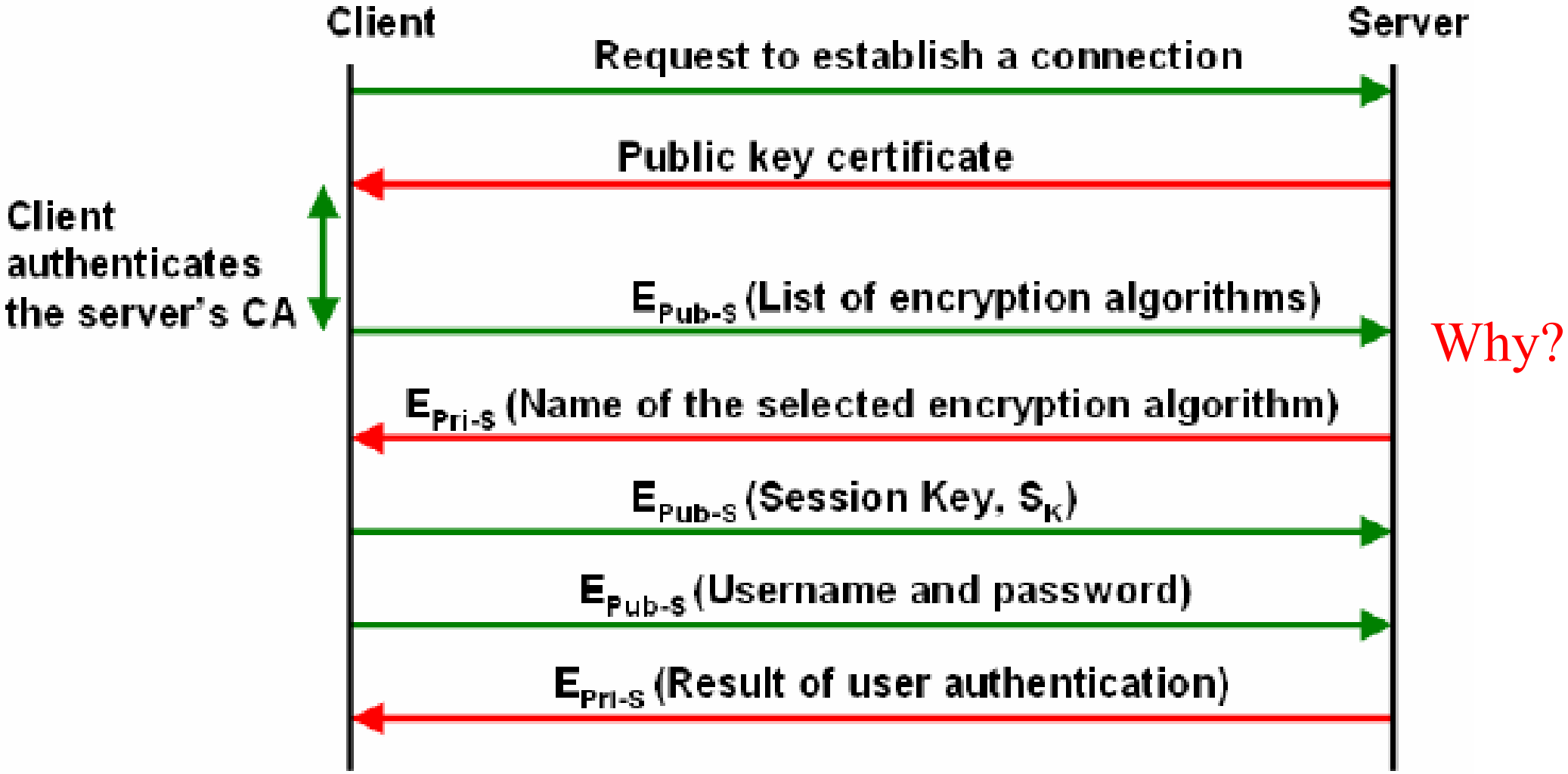
- **Biometric (variation)**
  - Verification?
  - Secure storage?
- **Smartcard**
  - Secure storage
  - Do some verification algorithms
- **Hardware-assisted (TPM, TEE, secure enclave...**
  - Secure storage
  - Do some secure algorithms

Network secure protocols

- Authentication
  - Public key (Certificate) approach
  - Prior secret-shared approach
- Agreement on cryptographic algorithms
- Key agreement
  - Diffie-hellman key exchange + an other factor
- Deployment
  - Chosent layer to implement
  - Chosen cipher to encrypt exchange data
- Some example secure protocols

Network secure protocols

Example: SSH protocols

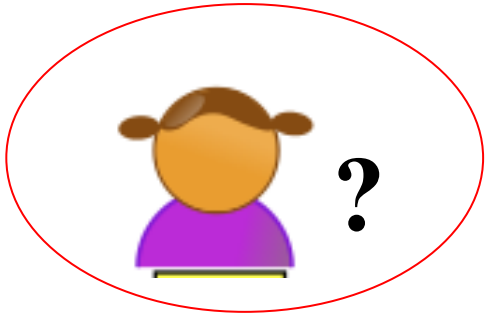


Why?

Network secure protocols

- Authentication
  - Public key (Certificate) approach
  - Prior secret-shared approach
- Key agreement
  - Diffie-hellman key exchange + an other factor
- Agreements of cryptographic algorithm
- Deployment
  - Chosent layer to implement
  - Chosen cipher to encrypt exchange data
- Some example secure protocols

Session Key agreement mechanism



Remote server

Server Certificate

SK_S

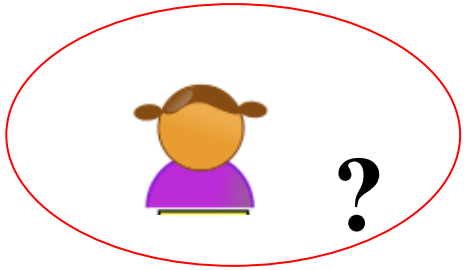
$$verify\ (certificate, PK_{root}) \Rightarrow PK_S$$

- How to send $(name, k_{shared})$ to server?

$$C = E_{AES}(ssk, name||pw)$$

$$D_{AES}(ssk, C) = name||pw$$

Session Key agreement mechanism



Remote server

Server Certificate

SK_S

$verify\ (certificate, PK_{root}) \Rightarrow PK_S$

- Using Diffie-Hellman key exchange + server public key

Example using ECC

- Select d_A
- Compute $Q_A = d_A G$

$name, C = E_{PK_S}(Q_A),$
$tag = h(Q_A||name)$ →

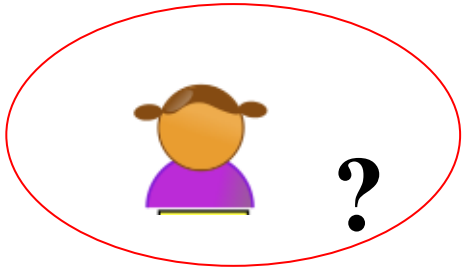
- $D_{SK_S}(C) = Q_A'$
- Verify $tag = h(Q_A'||name)$
- Select d_B
- Compute $Q_B = d_B G$

← Q_B
$tag_S = h(d_B Q_A||name)$

- Verify $tag_S = h(d_A Q_B||name)$

$$d_A Q_B = d_B Q_A = d_A . d_B G = ssk$$

Session Key agreement mechanism



Remote server

Server Certificate

SK_S

	UserName	HashedSaltedPwd	UserID
1	A-JayBibbins637	0x600FAD66D16A56AF3459D9C996DFF98B	10000
2	A-JayTorain976	0x4E30184141CB6CFF4120D6AC443CCE54	10001
3	AadamDobbin507	0xED2A51BD92E1EE8333314407817CD6F9	10002

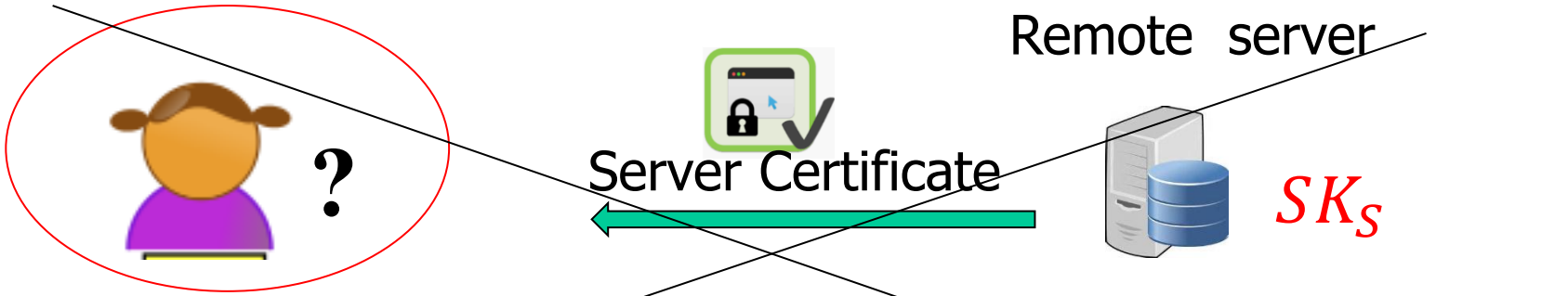
$$C = E_{AES}(ssk, name||pw)$$

$$D_{AES}(ssk, C) = name||pw$$

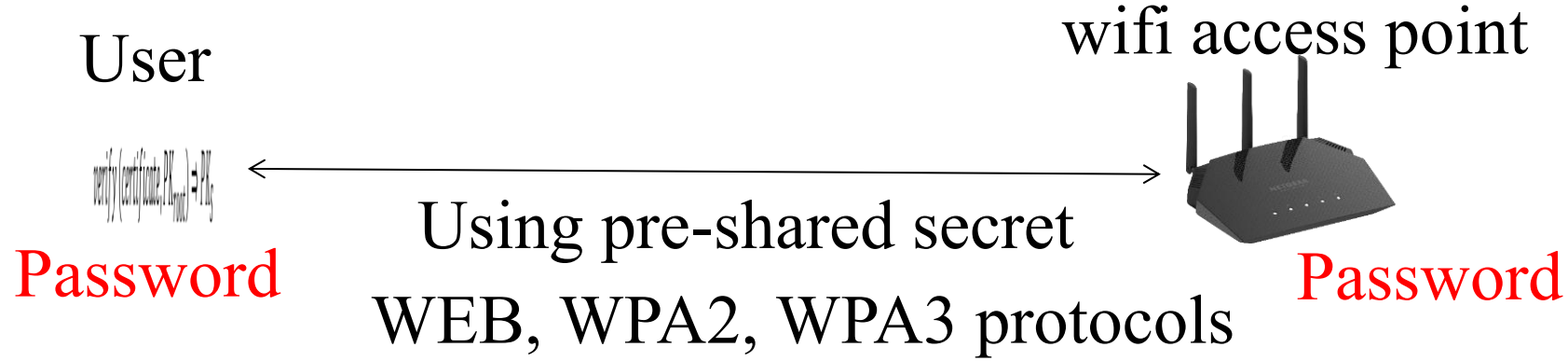
- Locate the row using name
- Check $h(pw')? = h(pw)$

→ authenticate the user

Session Key agreement mechanism

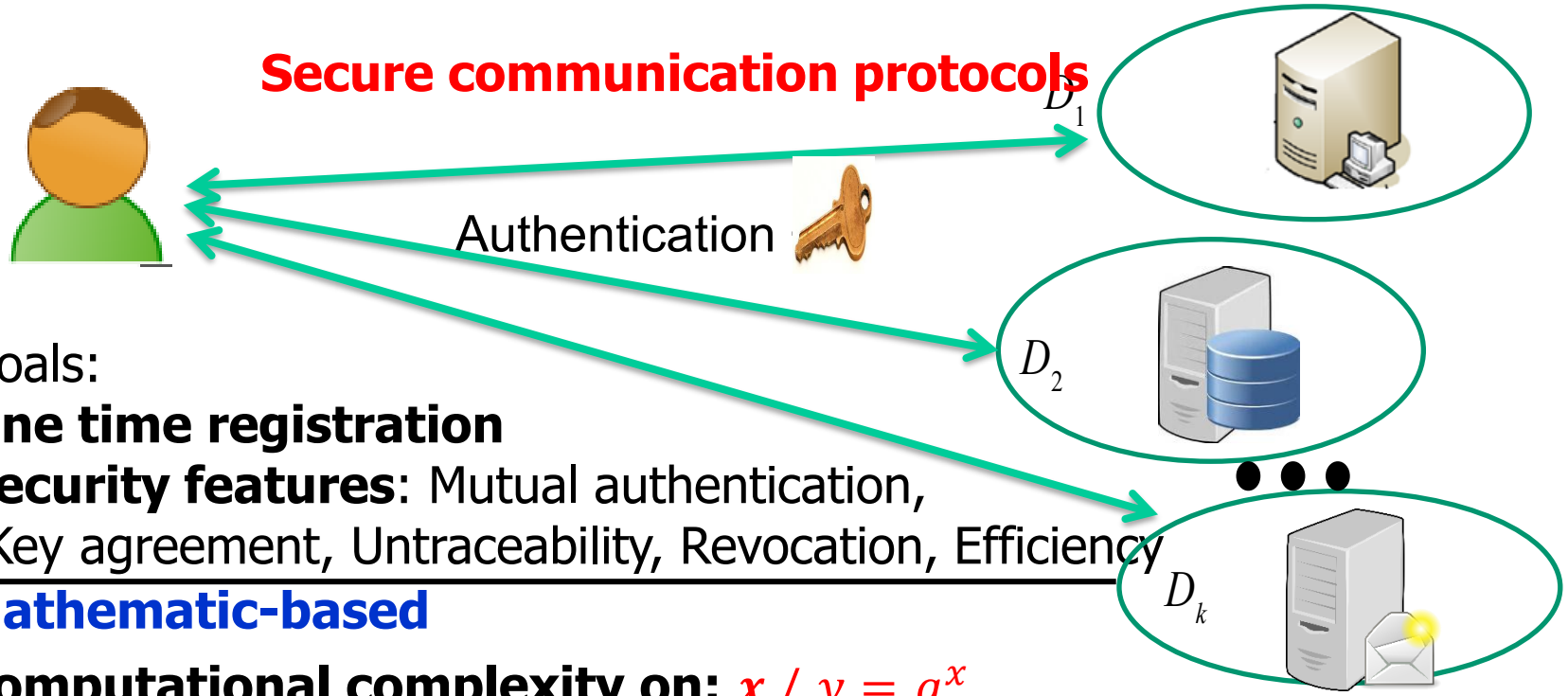


How authenticate each other if the server does not have certificate?



Certificateless AKA 1

Multi-server environments



- Goals:
  - **One time registration**
  - **Security features**: Mutual authentication, Key agreement, Untraceability, Revocation, Efficiency

---

❖ **Mathematic-based**

- **Computational complexity on:** $x / y = g^x$
- Hash Function; Random oracle model (Probability)

❖ **Authentication factors:** Passwords, smart card, biometric.

Chang, Chin-Chen, and **Nguyen, Ngoc-Tu**. "An Untraceable Biometric-Based Multi-server Authenticated Key Agreement Protocol with Revocation." Wireless Personal Communications 90.4 (2016): 1695-1715.

Certificateless AKA 2

➢ Goals:

- **Security features**: Mutual authentication, Key agreement, Untraceability Revocation, Efficiency
- **Provable security in ROR model**

❖ **Mathematic-based**

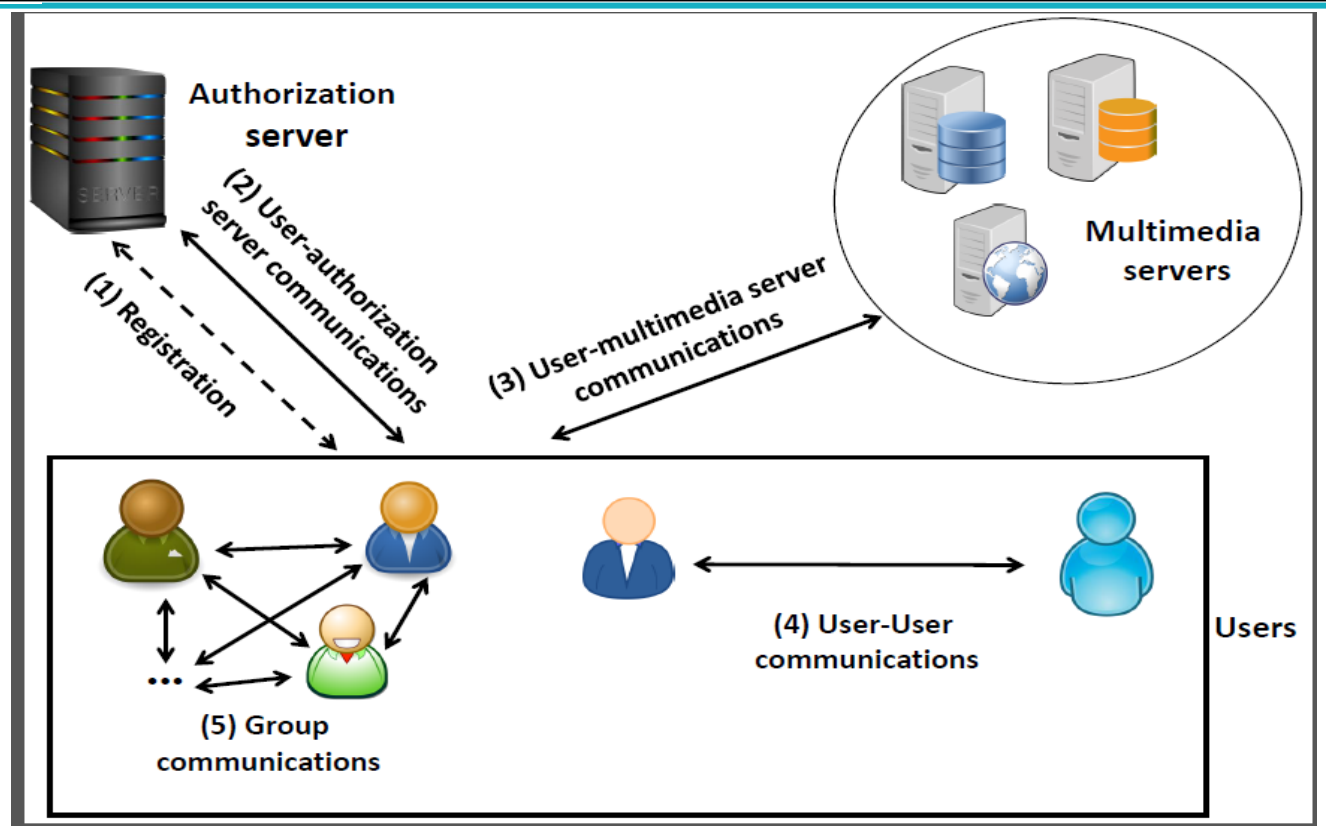
- Computational complexity on a addition group on Elliptic-curve cryptography: $\mathrm{d} / \mathrm{Q} = dG$

❖ **Authentication factors:** Passwords, smart card, biometric.

- Hash Function; Random oracle model (Probability)

Nguyen, Ngoc-Tu, and Chin-Chen Chang. "Untraceable biometric-based three-party authenticated key exchange for dynamic systems." Peer-to-Peer Networking and Applications 11.3 (2018): 644-663..

Certificateless AKA 3



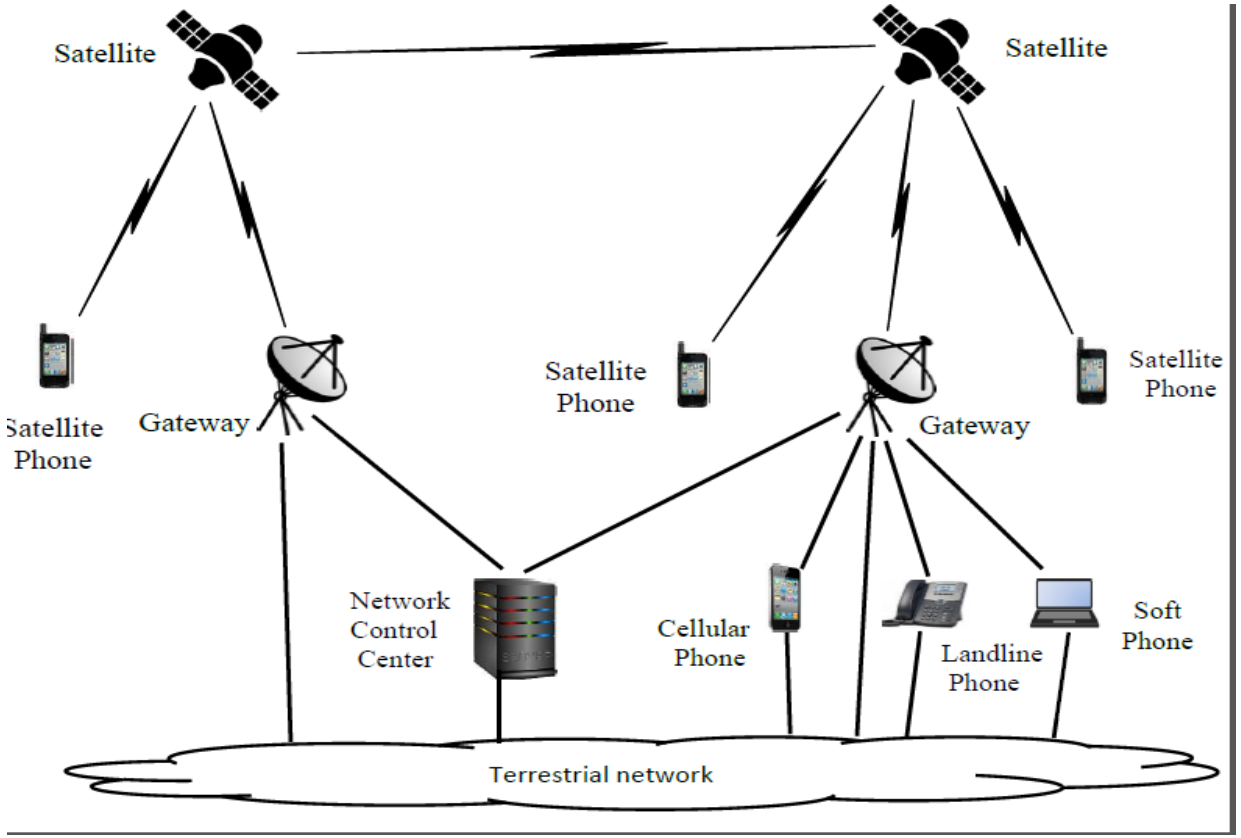
SIP Protocols

❖ Mathematic-based

- Computational complexity on a addition group on Elliptic-curve cryptography: $\mathrm{d} / \mathrm{Q} = dG$

Nguyen, Ngoc-Tu, and Chin-Chen Chang. "A biometric-based authenticated key agreement scheme for session initiation protocol in ip-based multimedia networks." Multimedia Tools and Applications (2018): 1-39.

Certificateless AKA 4



Satellite Mobile Protocols

❖ **Mathematic-based**

- Computational complexity on a addition group on Elliptic-curve cryptography: $\mathrm{d} / \mathrm{Q} = dG$

Nguyen, Ngoc-Tu, and Chang, Chin-Chen. "A Biometric-based Authenticated Key Agreement Protocol for User-to-user Communications in Satellite Mobile Networks.", Wireless Personal Communications. 2019 Aug 1;107(4):1727-58

Network secure protocols

- Authentication
  - Public key (Certificate) approach
  - Prior secret-shared approach
- Agreements of cryptographic algorithm
- Key agreement
  - Diffie-hellman key exchange + an other factor
- Deployment
  - Chosent layers to implement
  - Chosen cipher to encrypt exchange data
- Some example secure protocols

Deployment network secure protocols

- **Encryption** and **authentication** algorithms are building blocks of secure network protocols
  - Deploying cryptographic algorithms at different layers have different security effects
  - Where should we put the security protocol in the network architecture?

Deployment network secure protocols

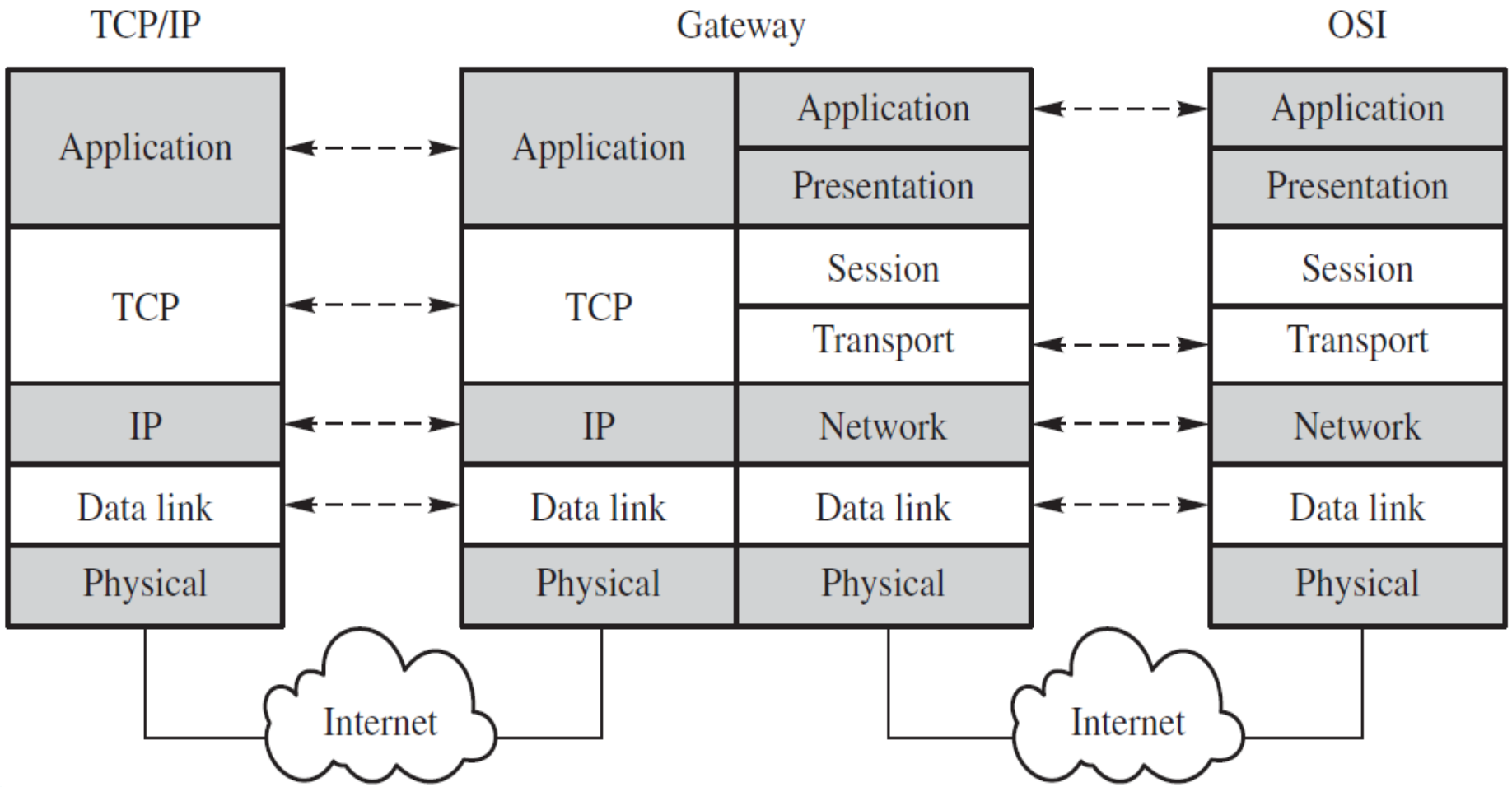
How to establish secure protocol?

1. Authentication?
2. Key agreement
3. Chosen cipher for exchage data

AES-128-CBC, SHA256,
AES-256-GCM, SHA3-512, …

Deployment network secure protocols

Where should we deploy the secure protocol?



Deployment network secure protocols

Logical (Software)

- Application
  - Web, Email
- Transport Layer
  - TCP, UDP
- Network Layer
  - IP

Physical (Hardware)

- Data Link Layer
  - Ethernet, 802.11
- Physical Layer

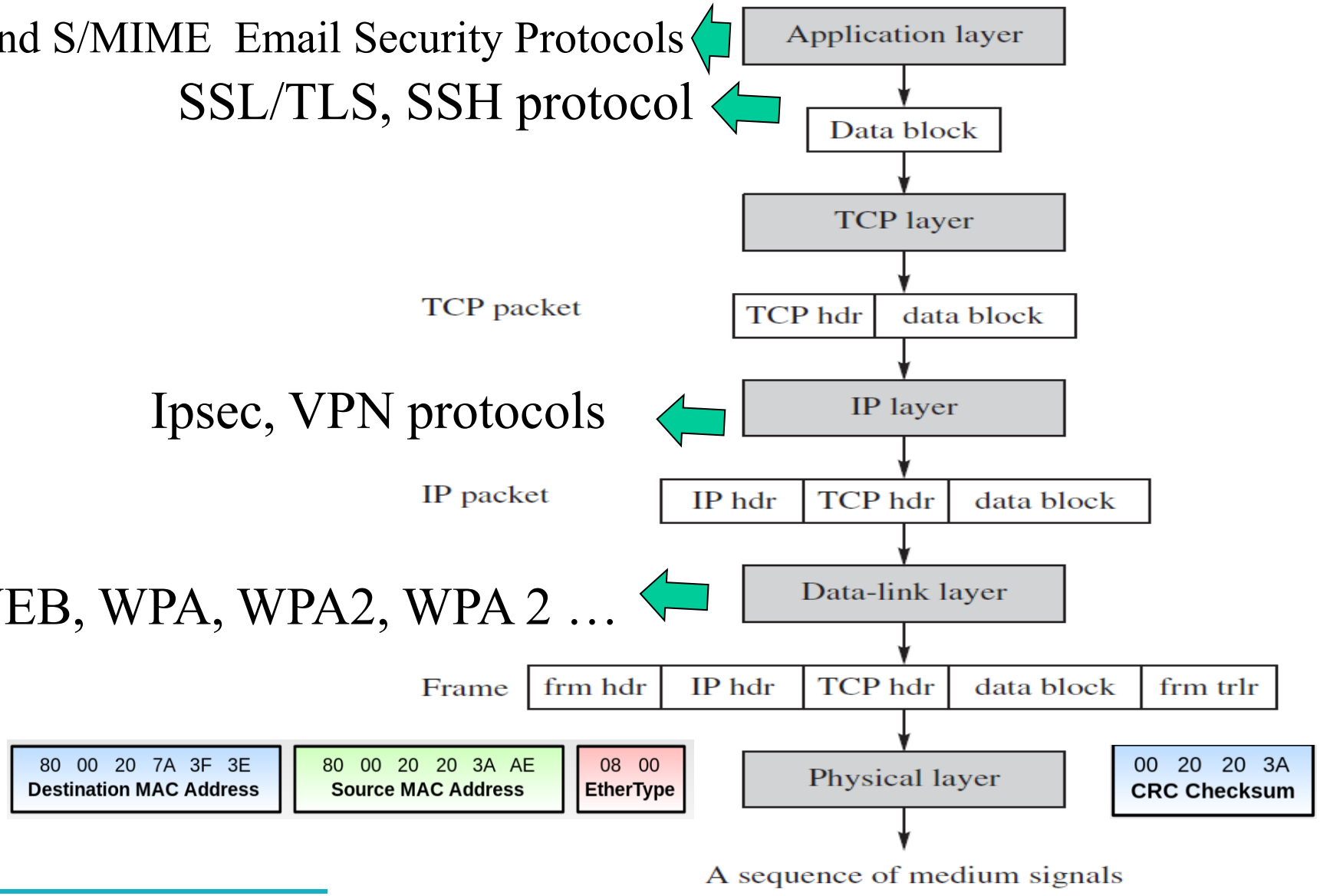
Deployment network secure protocols

PGP and S/MIME Email Security Protocols

SSL/TLS, SSH protocol

Ipsec, VPN protocols

WEB, WPA, WPA2, WPA 2 …



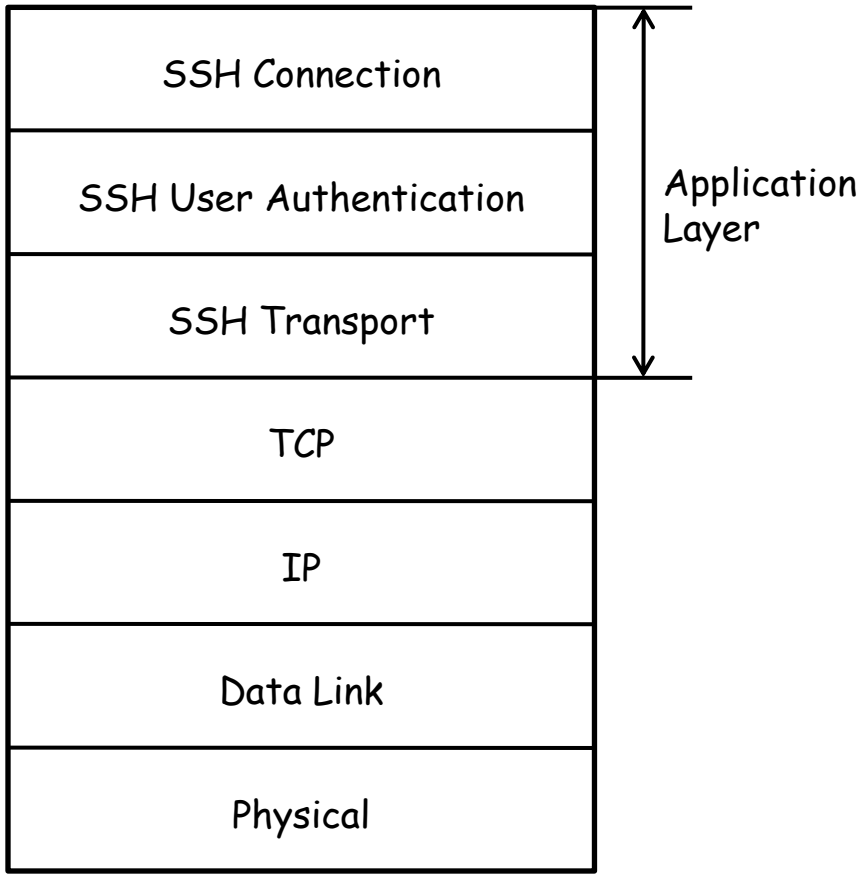
What Are the Pros and Cons?

- **Application Layer**
  - Provides end-to-end security protection
  - Intermediate nodes need not to decrypt data or check for signatures
  - Attackers may analyse traffic and modify headers
- **Transport Layer**
  - Provides security protections for TCP packets
  - No need to modify any application programs
  - Attackers may analyse traffic via IP headers

What Are the Pros and Cons?

- **Network Layer**
  - Provides node-to-node security protection
    - Transport mode: Encrypt payload only
    - Tunnel mode: Encrypt both header & payload; need a gateway
  - No need to modify any application programs
- **Data-link Layer**
  - Provides security protections for frames
  - No need to modify any application programs
  - Traffic analysis would not yield much info

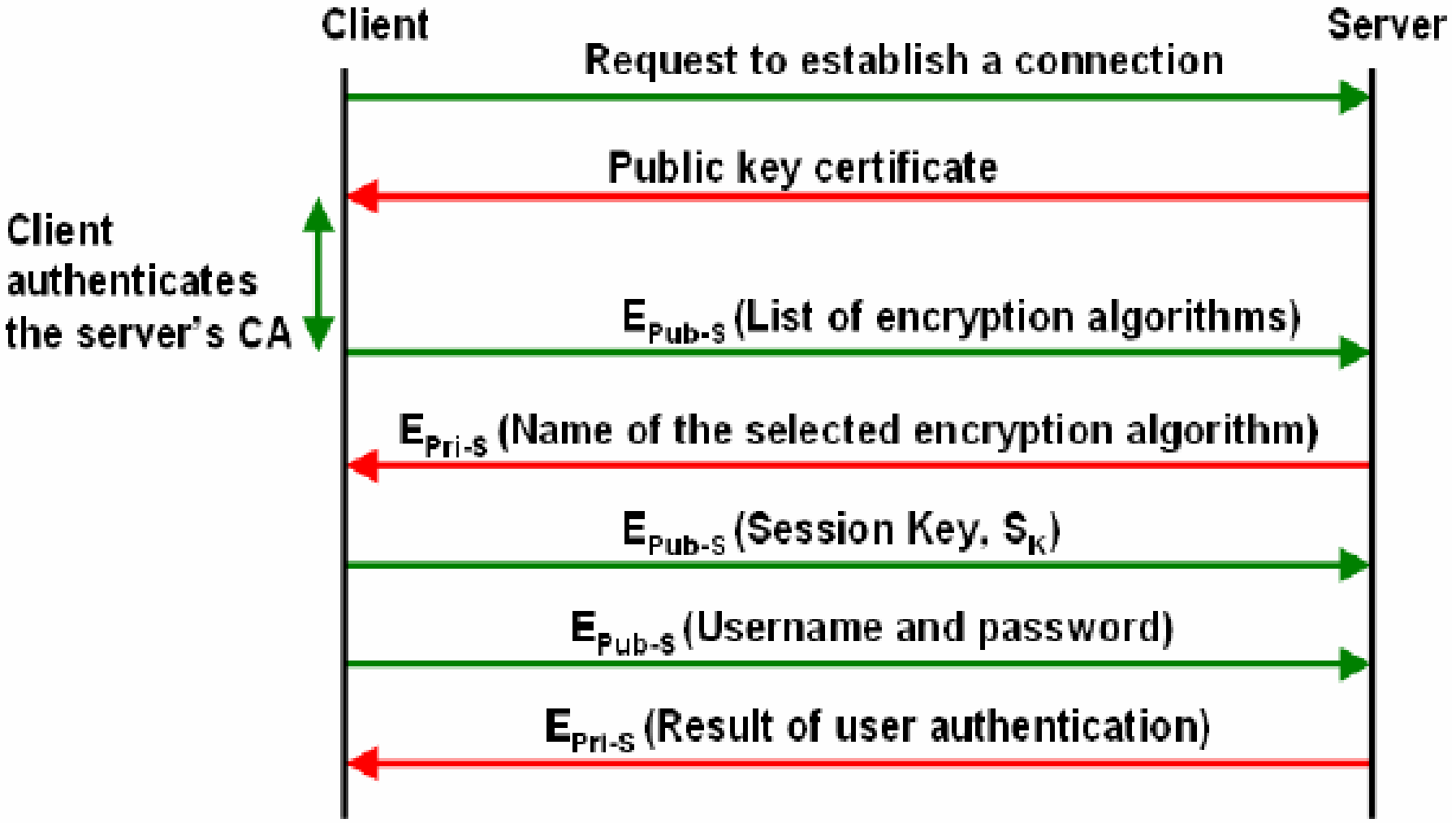
SSH protocol



SSH architecture

- SSH Connection:
  - Sets up multiple channels for different applications in a single SSH connection
- SSH User Authentication:
  - Authenticate user to server
  - Using password or PKC
- SSH Transport
  - Handles initial setup: server authentication, and key exchange
  - Set up encryption and compression algorithms

SSH protocol



SSL/TLS

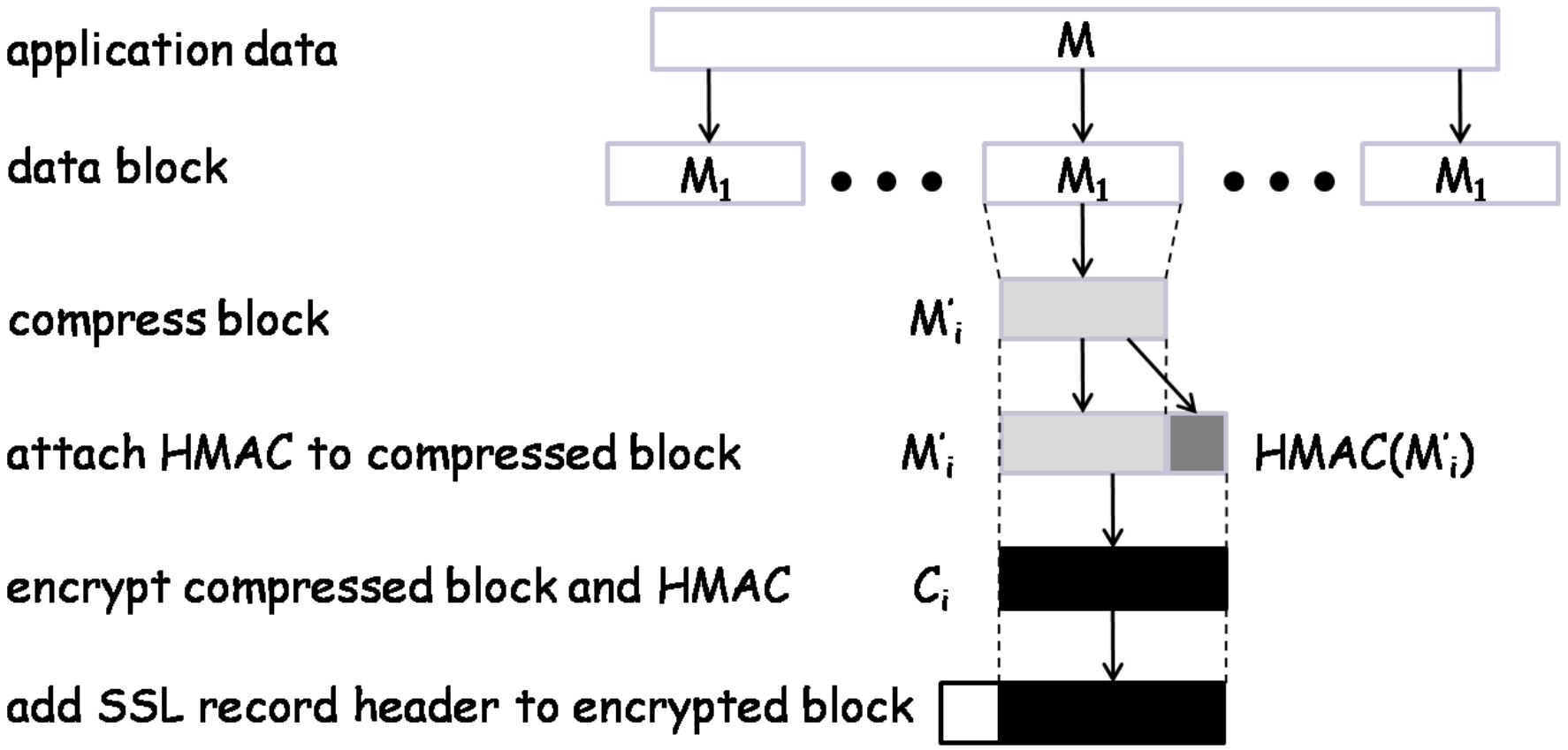
- Secure Socket Layer Protocol (SSL)
  - Designed by Netscape in 1994

SSL and TLS protocols

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecated in 2021 (RFC 8996)[8][9][10]
TLS 1.1	2006	Deprecated in 2021 (RFC 8996)[8][9][10]
TLS 1.2	2008	
TLS 1.3	2018	

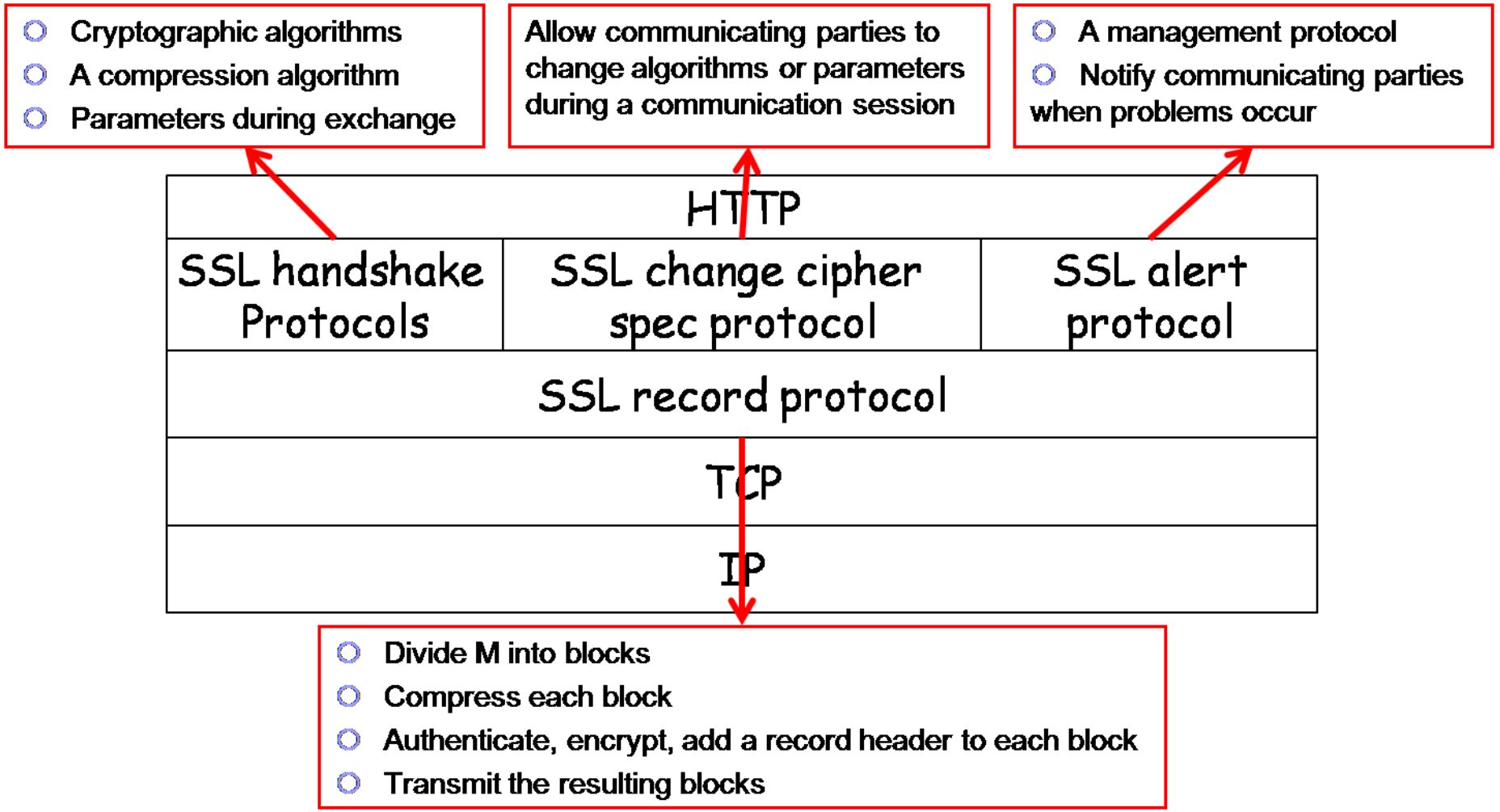
https://en.wikipedia.org/wiki/Transport_Layer_Security

TLS Record Protocol Diagram



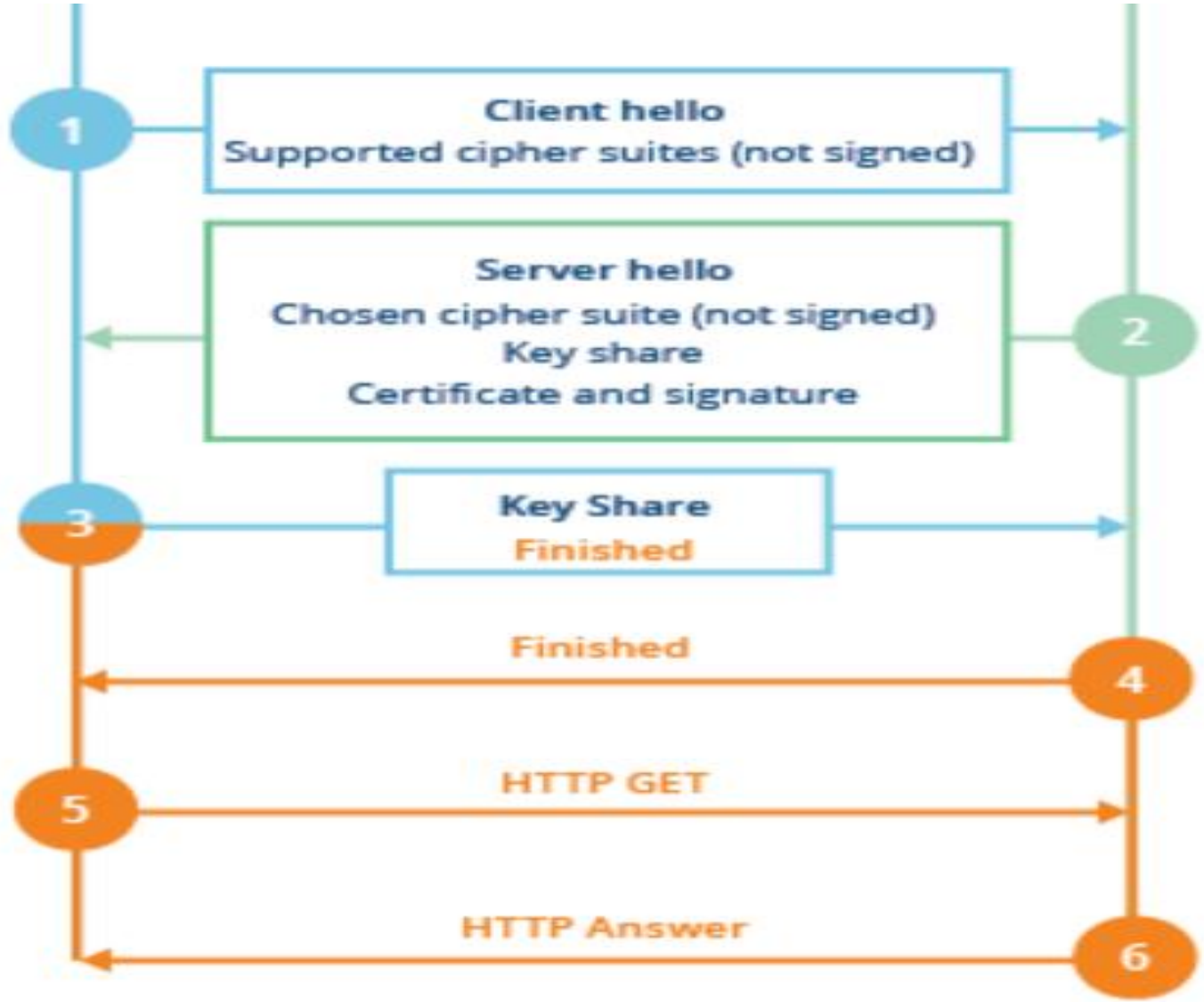
SSL record protocol

TSL Structure



TSL protocol v 1.2

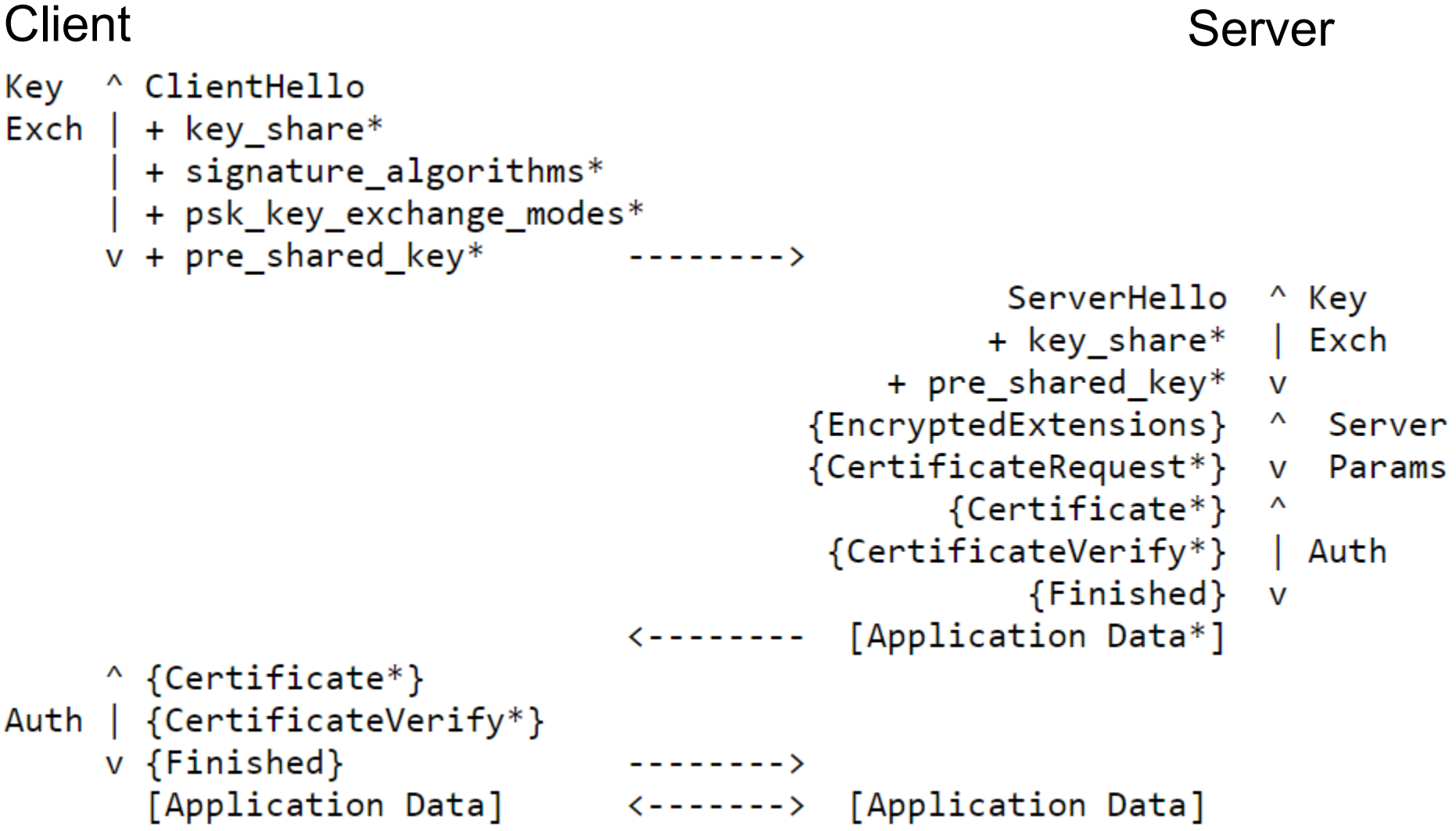
Client



Server

(EC)DHE

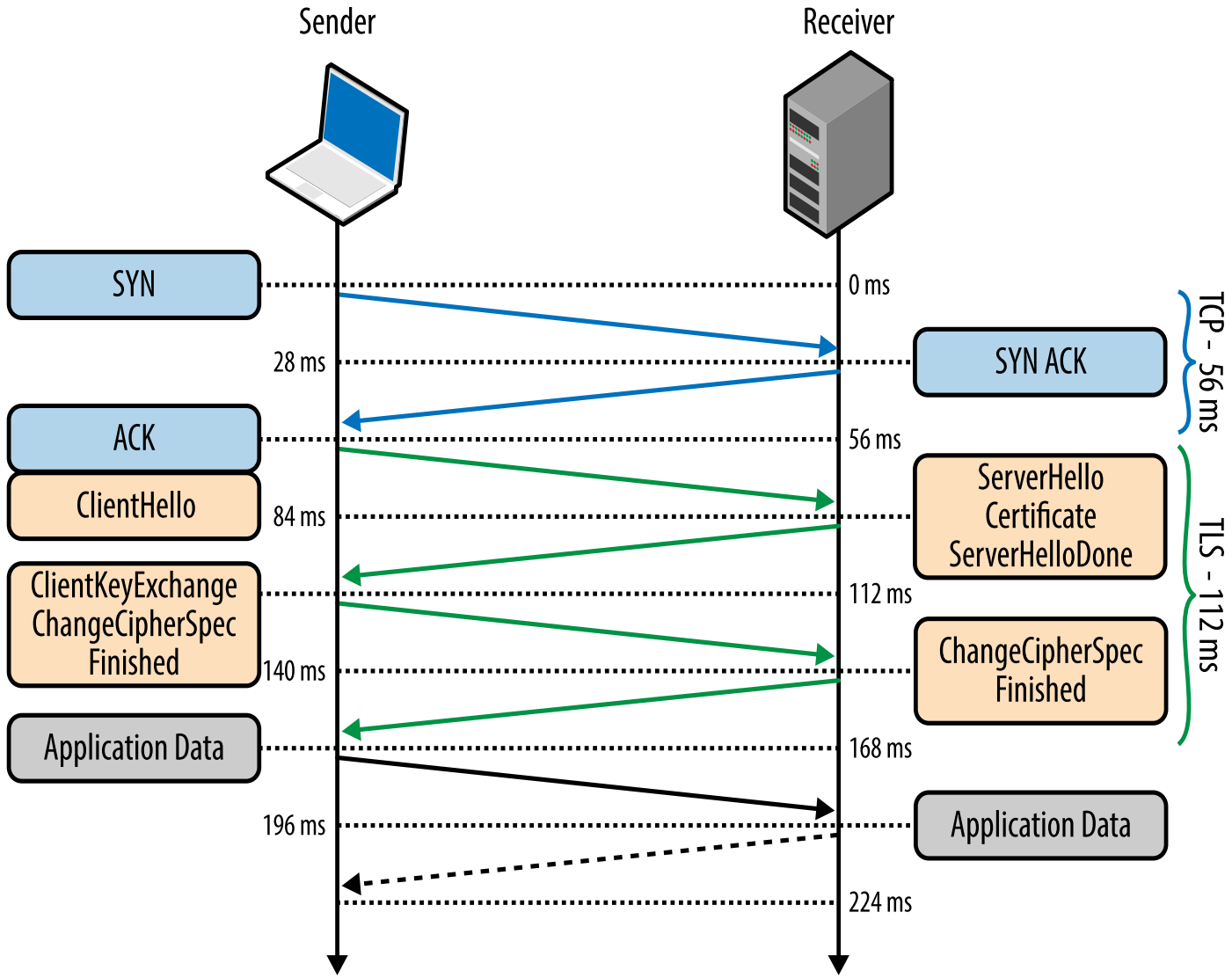
TSL protocol v 1.3



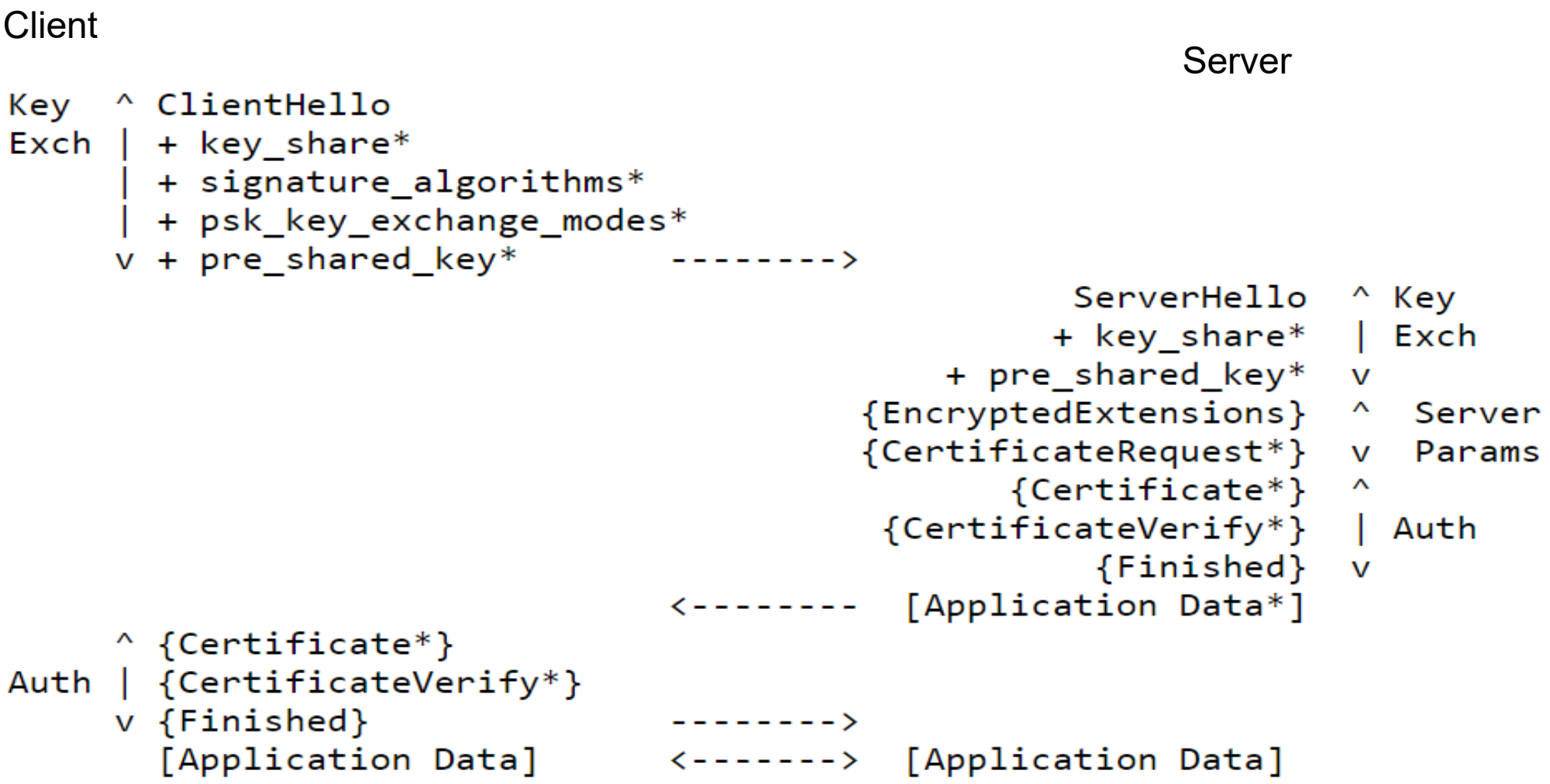
https://datatracker.ietf.org/doc/html/rfc8446

TSL protocol v 1.2

- Authentication: digital certificate;
- Key agreement(ex. ECDH)
- Cryptographic algorithm negotiation (ciphers, MAC)



TSL protocol v 1.3

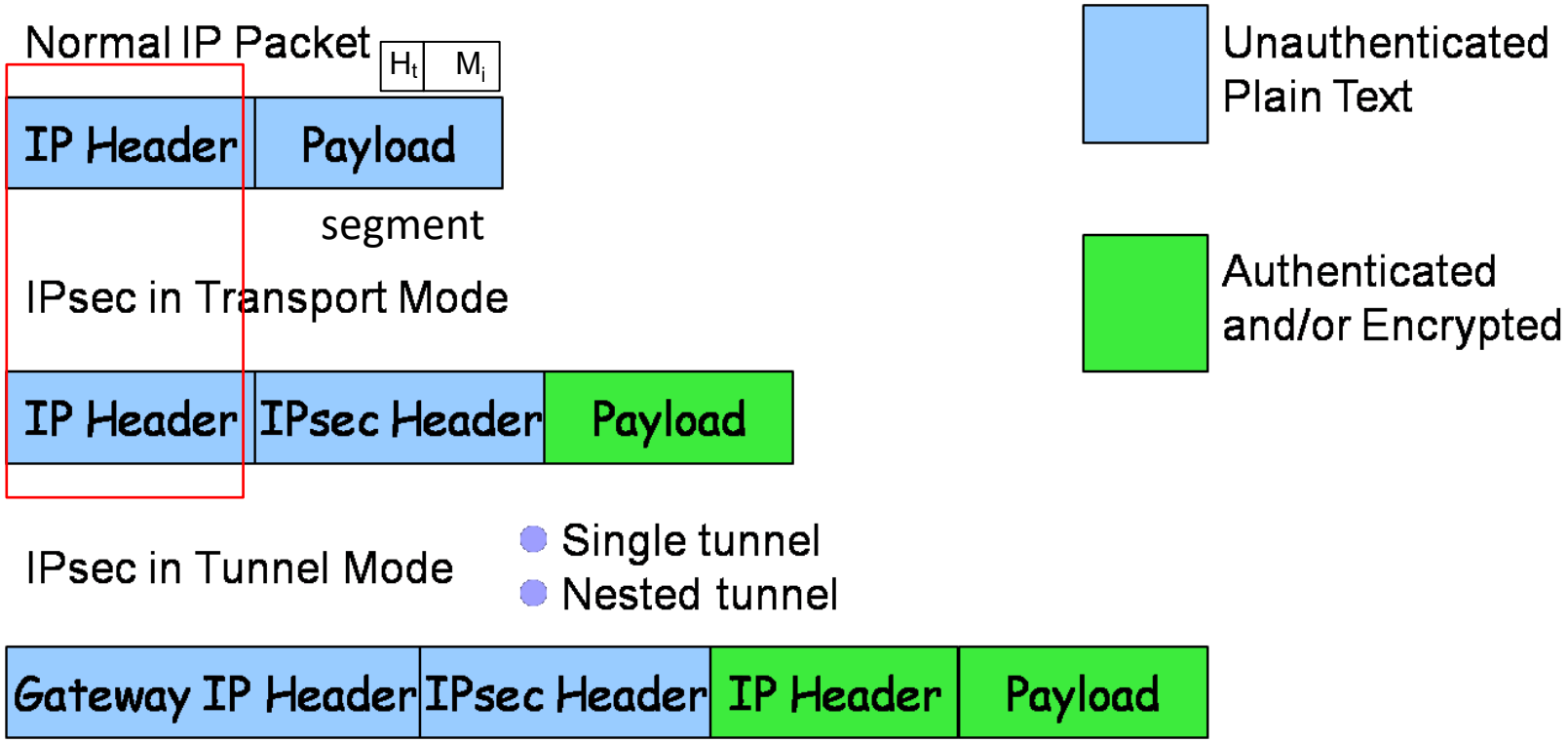


https://datatracker.ietf.org/doc/html/rfc8446

IPsec: Network-Layer Protocol

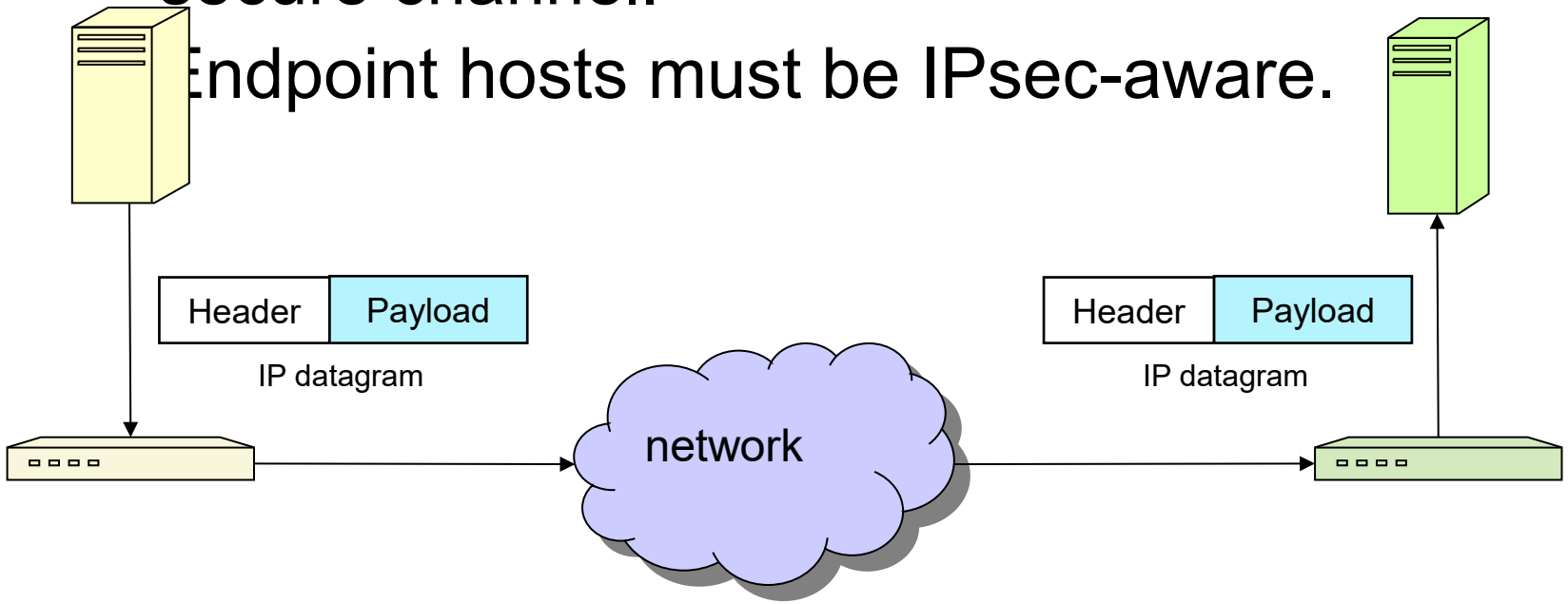
- IPsec encrypts and/or authenticates IP packets
- It consists of three protocols:
  - Authentication header (AH)
    - To authenticate the origin of the IP packet and ensure its integrity
    - To detect message replays using sliding window
  - Encapsulating security payload (ESP)
    - Encrypt and/or authenticate IP packets
  - Internet key exchange (IKE)
    - Establish secret keys for the sender and the receiver
- Runs in one of two modes:
  - Transport Mode
  - Tunnel Mode (requires gateway)

IPsec Packet Layout

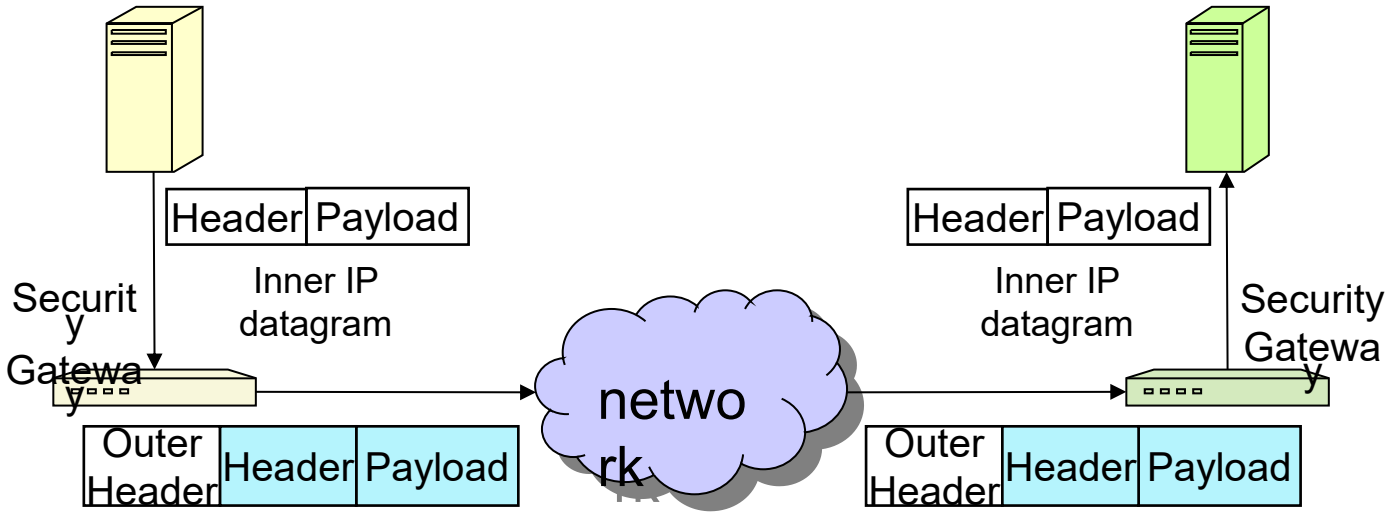


IPsec Transport Mode

- Host-to-host (end-to-end) security:
  - IPsec processing performed at endpoints of secure channel.
  - Endpoint hosts must be IPsec-aware.

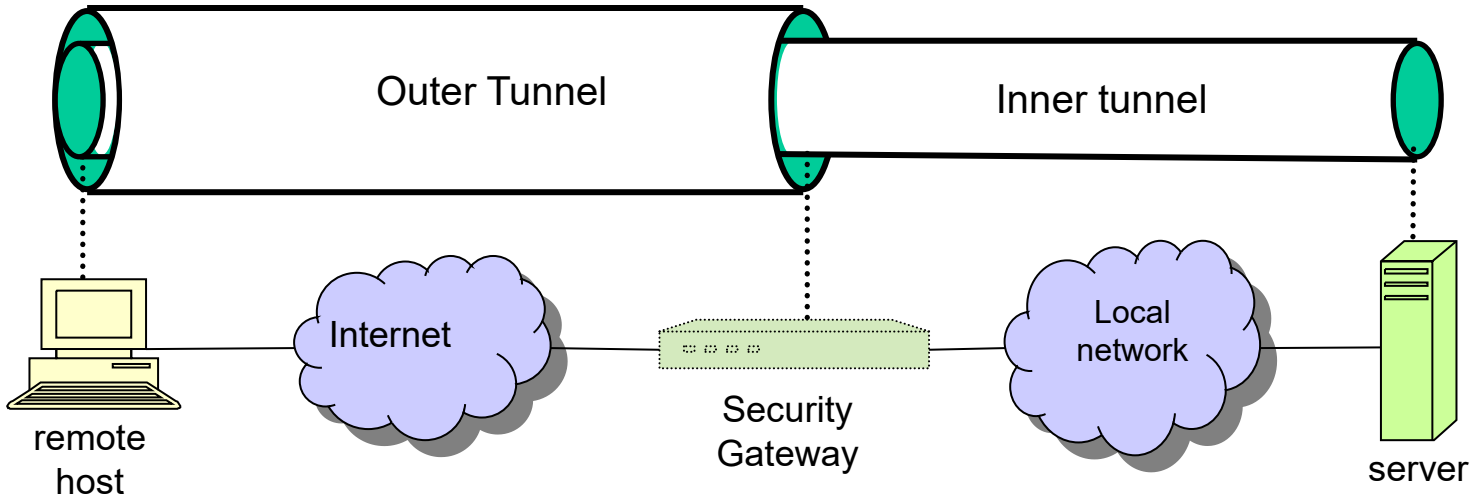


IPsec Tunnel Mode



Remote Host to Internal Server

- Remote host has Internet access to gateway, then gains access to server behind gateway.
- Traffic to server protected in inner tunnel.
- Outer tunnel protects inner traffic over Internet.



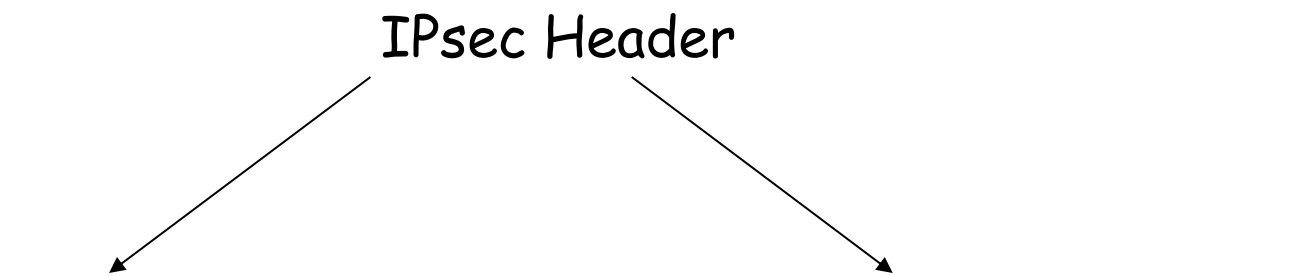
IPsec Security Associations

- If Alice wants to establish an IPsec connection with Bob, the two parties must first negotiate a **set of keys and algorithms**
- The concept of **security association** (SA) is a mechanism for this purpose
- An SA is formed between an initiator and a responder, and lasts for one session
- An SA is for encryption or authentication, but not both.
- If a connection needs both, it must create two SAs, one for encryption and one for authentication

SA Components

- Three parameters:
  - Security parameters index (SPI)
  - IP destination address
  - Security protocol identifier
- Security Association Database (SAD)
  - Stores active SAs used by the local machine
- Security Policy Database (SPD)
  - A set of rules to select packets for encryption / authentication
- SA Selectors (SAS)
  - A set of rules specifying which SA(s) to use for which packets

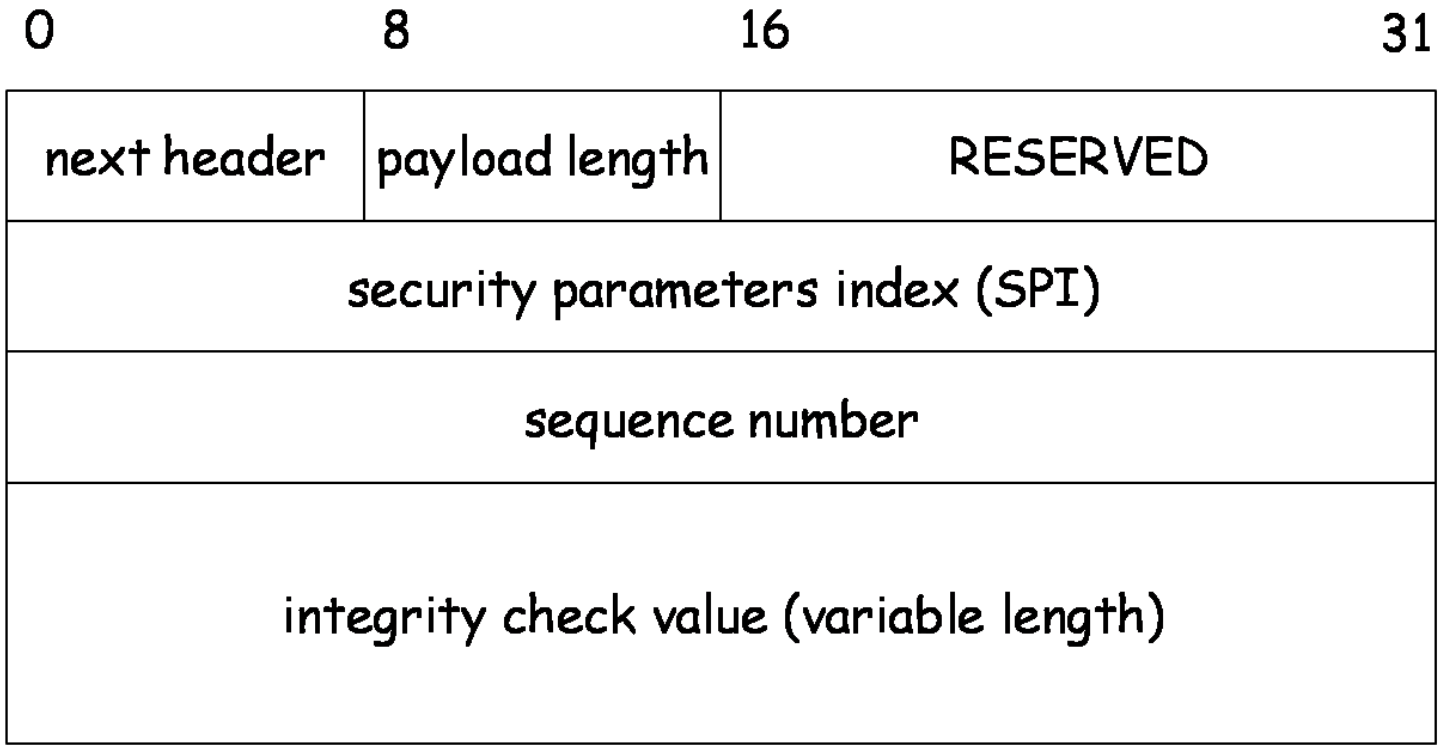
IPsec Header



Authentication and Encryption use separate SAs

- Data confidentiality
- Data origin authentication, data integrity checking, and replay protection (optional)

Authentication Header



Resist Message Replay Attack

Sequence number is used with a sliding window to thwart message replay attacks

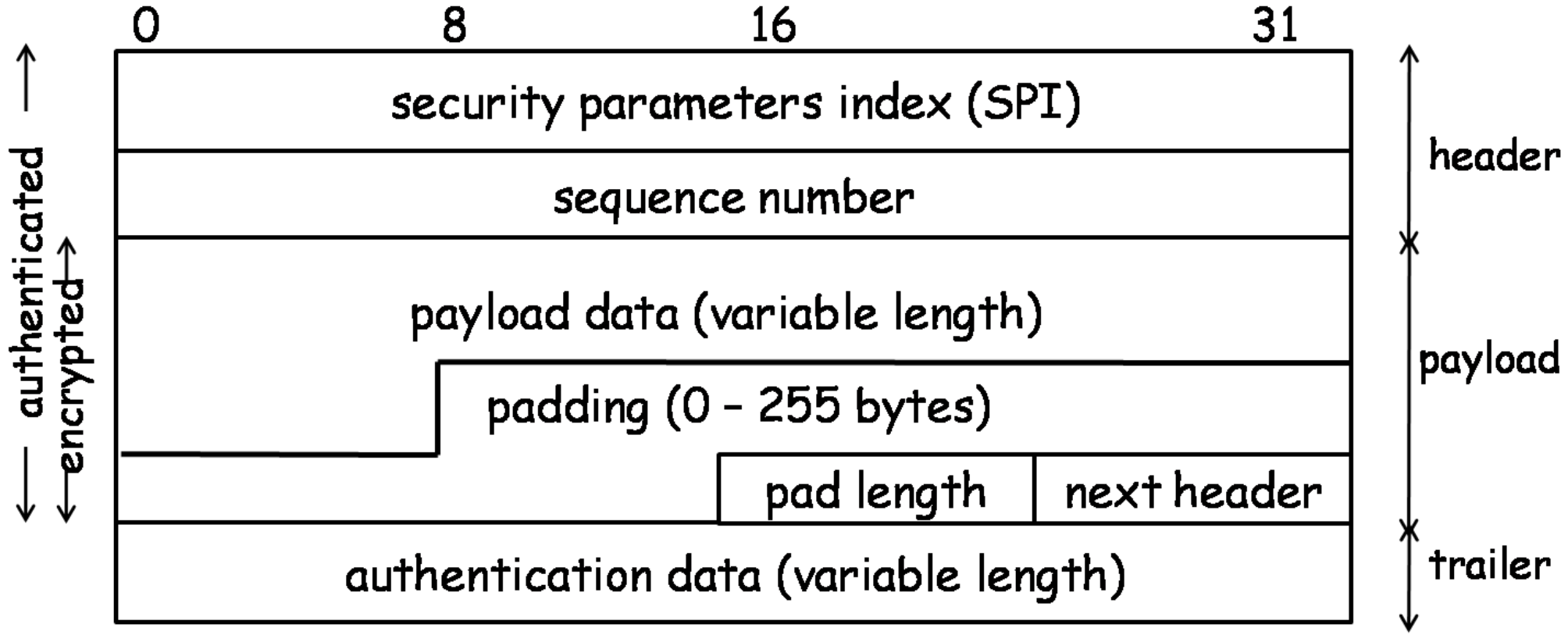
Given an incoming packet with sequence # s, either

s in A – It's too old, and can be discarded

s in B – It's in the window. Check if it's been seen before

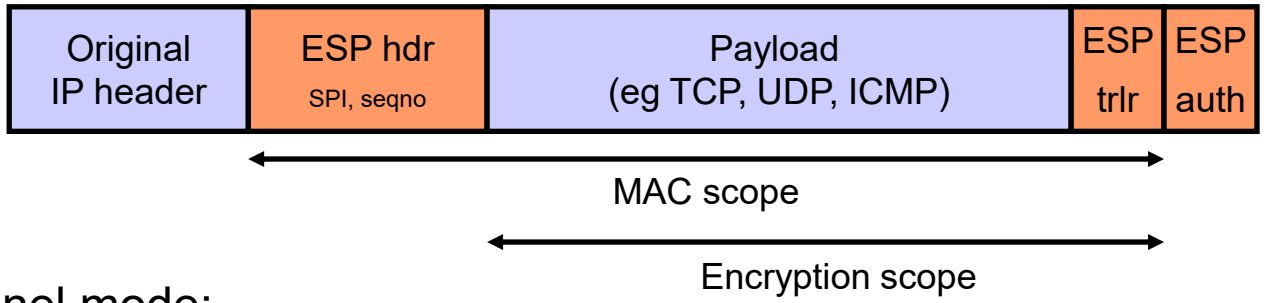
s in C – Shift the window and act like case B

Encapsulated Security Payload

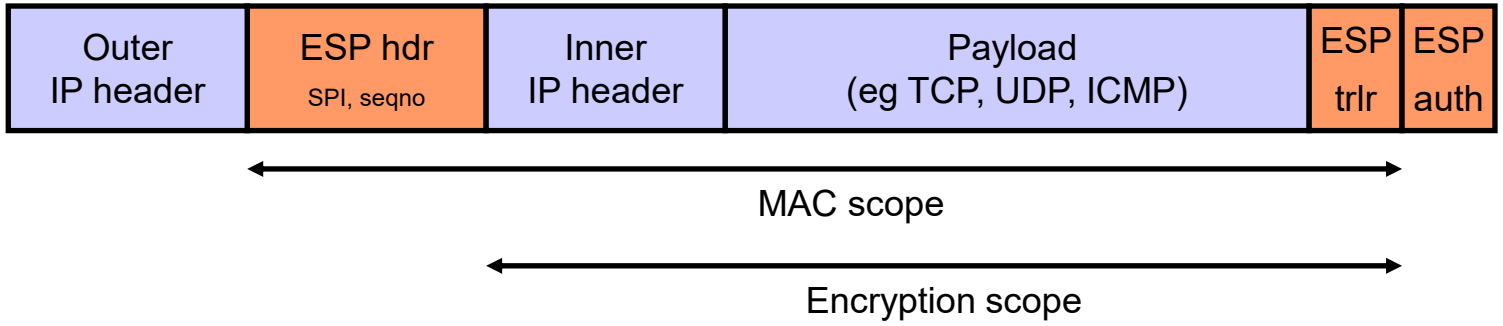


ESP Protocol – Transport & Tunnel

ESP in transport mode:



ESP in tunnel mode:



Key Determination and Distribution

- Oakley key determination protocol (KDP)
  - Diffie-Hellman Key Exchange
    + authentication & cookies
  - Authentication helps resist man-in-the-middle attacks
  - Cookies help resist clogging attacks
  - Nonce helps resist message replay attacks

Clogging Attacks

- A form of denial of service attacks
- Attacker sends a large number of public key Y_i in crafted IP packets, forcing the victim's computer to compute secret keys $K_i = Y_i^X \bmod p$ over and over again
  - Diffie-Hellman is computationally intensive because of modular exponentiations
- Cookies help
  - Before doing computation, recipient sends a cookie (a random number) back to source and waits for a confirmation including that cookie
  - This prevents attackers from making DH requests using crafted source addresses

ISAKMP

- ISAKMP: Internet Security Association and Key Management Protocol
  - Specifies key exchange formats
  - Each type of payload has the same form of a payload header

64-bit initiator's cookie				
64-bit responder's cookie				
8-bit next payload	4-bit major ver	4-bit minor ver	8-bit exchange type	8-bit flags
32-bit message ID				
32-bit length				

ISAKMP header

ISAKMP Payload Types

- *SA*: for establishing a security association
- *Proposal*: for negotiating an SA
- *Transform*: for specifying encryption and authentication algorithms
- *Key-exchange*: for specifying a key-exchange algorithm
- *Identification*: for carrying info and identifying peers
- *Certificate-request*: for requesting a public-key certificate
- *Certificate*: contain a public-key certificate
- *Hash*: contain the hash value of a hash function
- *Signature*: contain the output of a digital signature function
- *Nonce*: contain a nonce
- *Notification*: notify the status of the other types of payloads
- *Delete*: notify the receiver that the sender has deleted an SA or SAs