CSE 3231
Computer Networks

Secure Data Communications

William Allen, PhD

Spring 2022

Transmitting Data Securely

- One important use of cryptography is enabling the secure transmission and reception of data over a network
  - secure emails and messaging, file transfers, web connections, data sharing, etc.
- Specific protocols have been designed to support secure communications
  - SSH, HTTPS, TLS/SSL, S-MIME, etc.

Creating Secure Communication

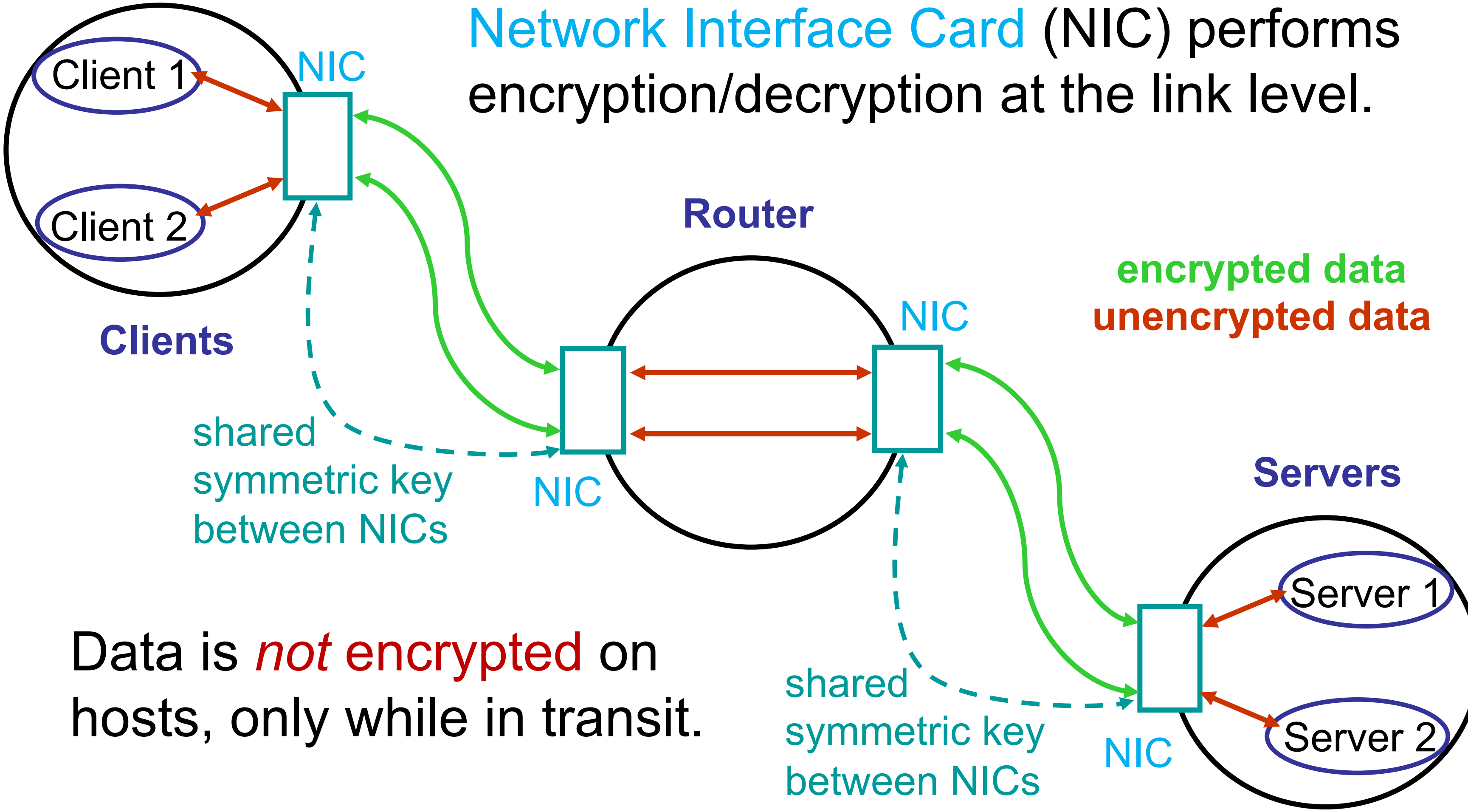
- To avoid sharing keys for each data file, several different approaches have been created to support secure communications
  - data can be encrypted as it travels between two computers over a direct link
  - data can be encrypted by an application and decrypted when it is received by another
  - virtual "*tunnels*" allow all communications between two computers to be encrypted

Encryption

- Link encryption – at lowest network layers
  – encrypts Ethernet payload, but not the header
  – encryption across direct links between nodes
  – requires compatible hardware at each end
  – key shared by network interfaces at nodes
- End-to-End encryption – at higher layers
  – only data is encrypted, routing is not affected
  – application encrypts data, network transmits
  – users/applications control keys

Link-Level Encryption

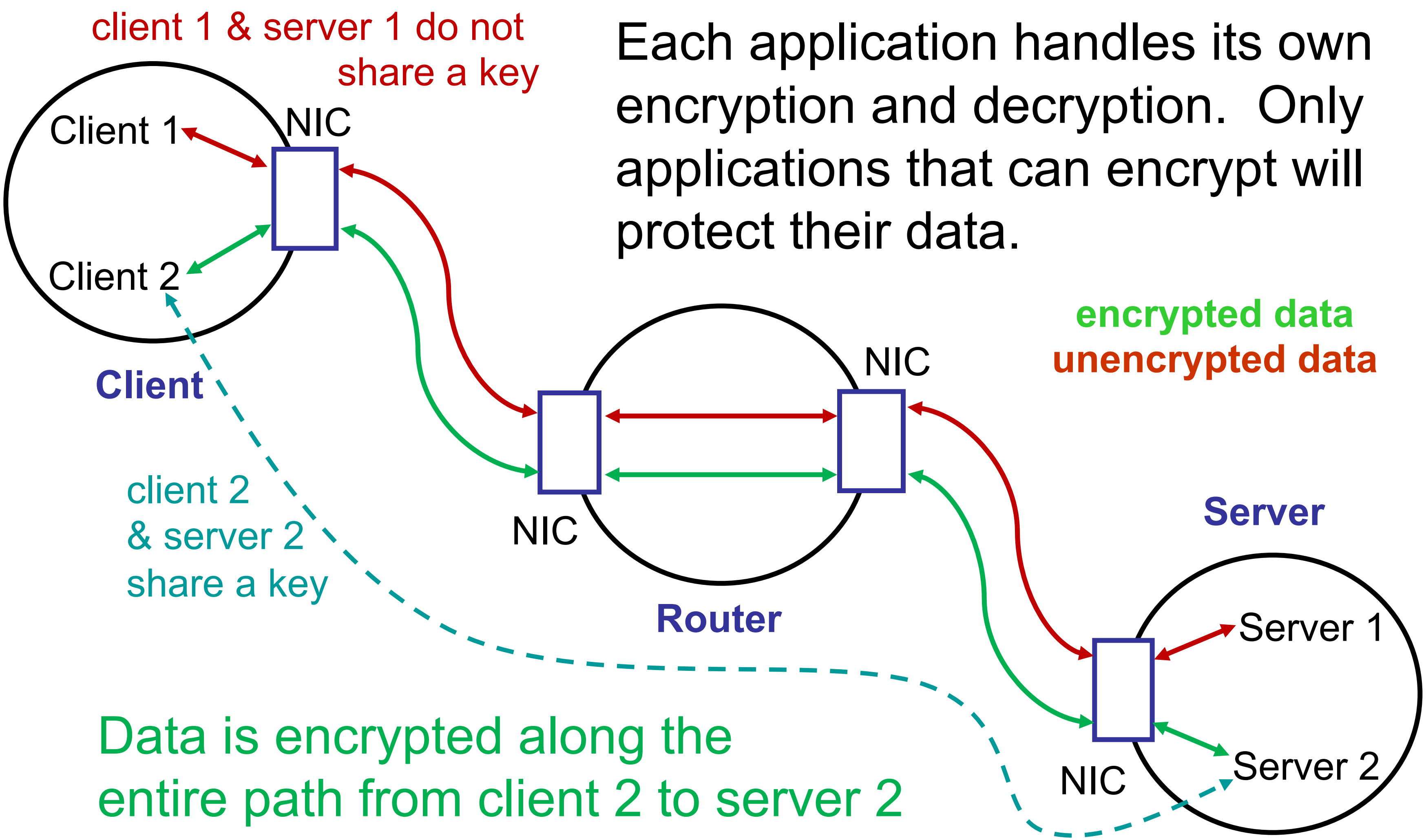
Network Interface Card (NIC) performs encryption/decryption at the link level.



Data is *not* encrypted on hosts, only while in transit.

End-to-End Encryption

Each application handles its own encryption and decryption. Only applications that can encrypt will protect their data.



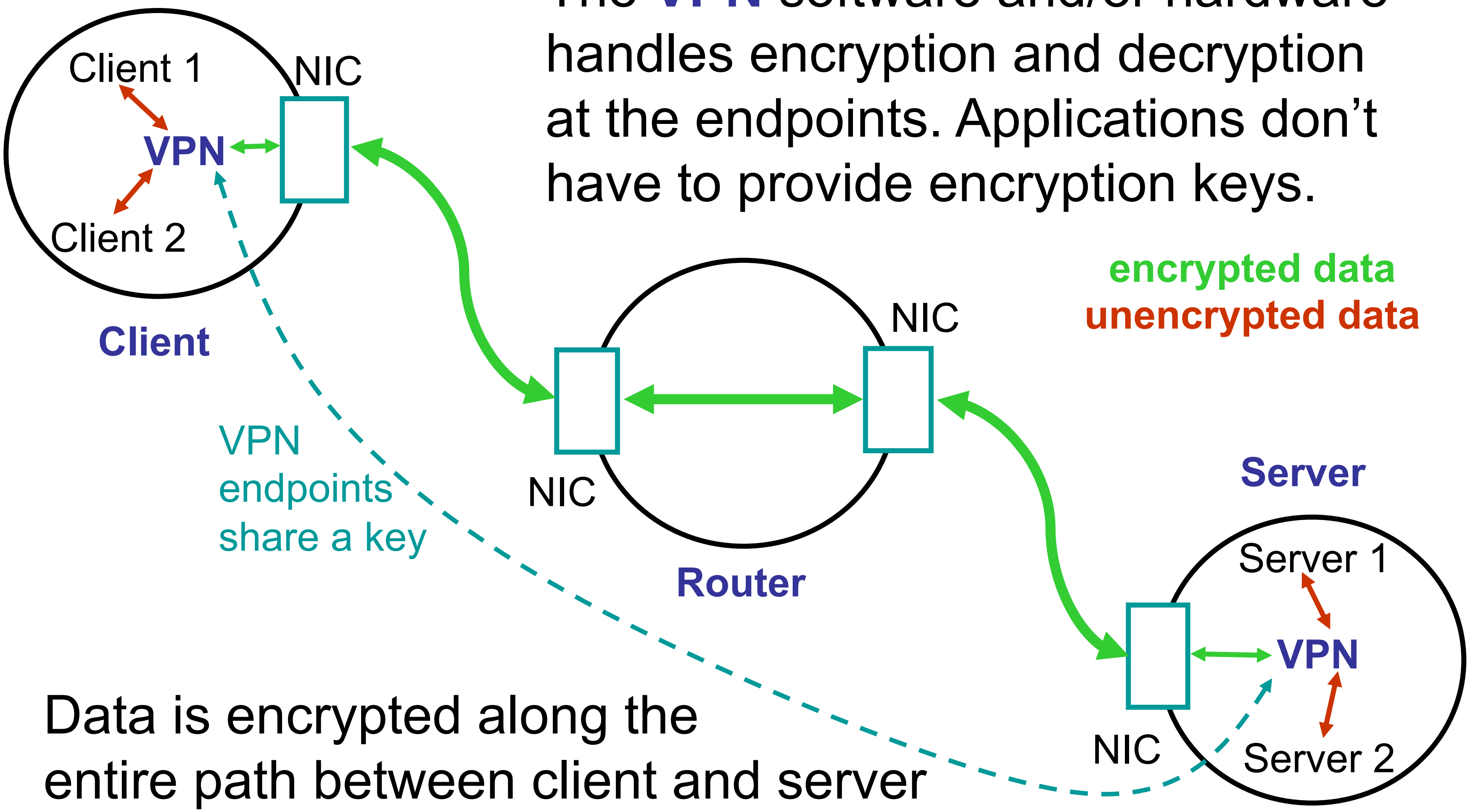
Data is encrypted along the entire path from client 2 to server 2

Virtual Private Network (VPN)

- Connects a remote computer to a secure network by creating an *encrypted tunnel* between the computer and the network
- The remote host must be authenticated by the secure network it connects to
- Protects data in transit to/from network
- Applications on host send data through the tunnel, they don't have to add encryption

VPN Encryption

The **VPN** software and/or hardware handles encryption and decryption at the endpoints. Applications don't have to provide encryption keys.



Data is encrypted along the entire path between client and server

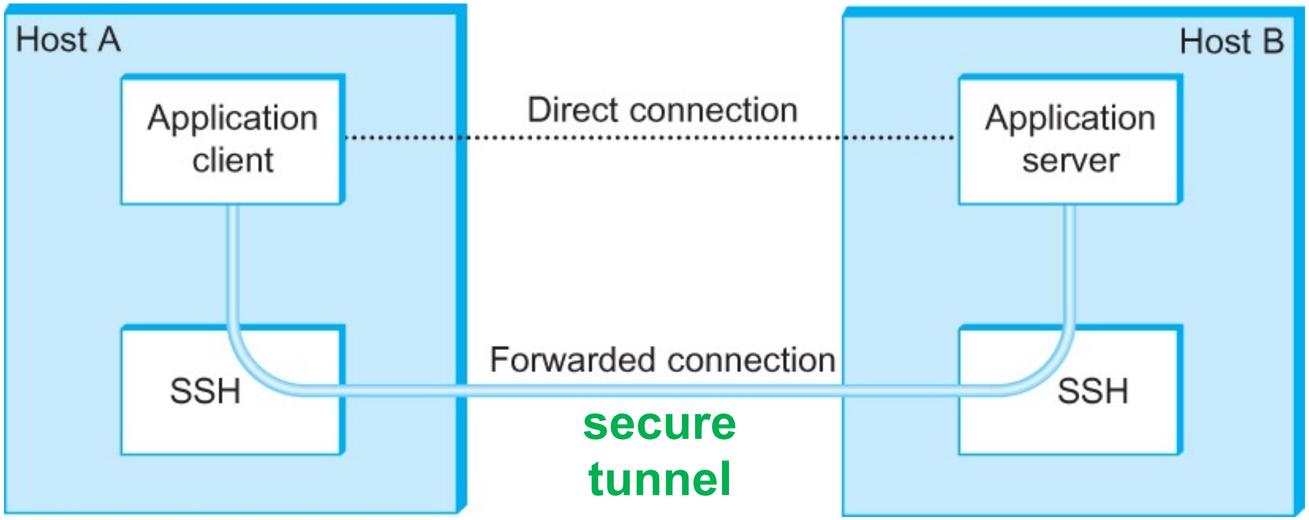
Insecure Protocols

- Most of the original network application protocols (email, Web, telnet, ftp, etc.) transferred data in *unencrypted* form
- Two main approaches were employed to provide data protection:
  - create an encrypted connection and tunnel the unencrypted connection inside it
  - revise the protocol to add an encrypted version, leaving the old protocol intact
- Some applications use both methods

Secure Shell (SSH)

- The telnet and rlogin protocols allowed users to connect to remote computers as if they were logged in directly at the console
  - However, they sent login/password and user commands over the network in plain text
- The Secure Shell (SSH) protocol was developed to provide a secure remote login service by creating an encrypted connection before transferring any data or commands
  - SSH can also be used to create a *secure tunnel* between hosts

Secure Shell (SSH)



SSH can use port forwarding to secure other TCP-based applications.

It creates a secure "tunnel" on different ports from the normal ports the application uses

Secure Sockets Layer (SSL)

- SSL supports secure end-to-end network connections for TCP applications
  - first used in the Netscape browser in 1995
  - updated over time to fix flaws and vulnerabilities
  - SSL provides a layer between the application and the TCP layer, encrypting all data in the TCP packet using a shared session key
  - the approach the protocol was based on was good, but it relied on obsolete or vulnerable components (e.g., MD5 and SHA-1)
  - *replaced by the Transport Layer Security* (TLS) protocol in the 2000's and deprecated in 2015

Transport-layer security (TLS)

- Widely deployed security protocol above the transport layer
  - supported by browsers, web servers, etc.
- TLS provides:
  - authentication:
    - established by public key certificates
  - confidentiality:
    - creating a shared session key that is exchanged during the configuration stage
  - integrity:
    - verifying through cryptographic hash values included with messages

Transport-layer security (TLS)

Built upon well-known components that have already been widely used:

- *handshake*: Alice, Bob use their public-key certificates and private keys to authenticate each other and to exchange or create the shared secret
- *key derivation*: Alice, Bob use a shared secret to derive set of session keys
- *data transfer*: stream data transfer, data is seen as a series of records, not just one-time transactions

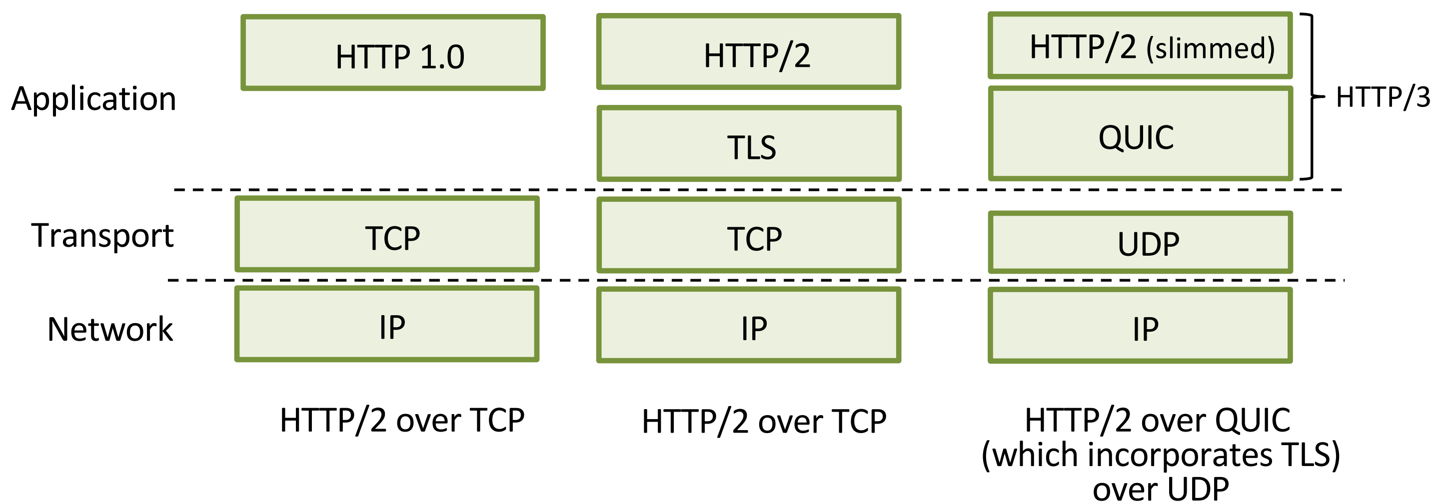
Transport-layer security (TLS)

Built upon well-known components that have already been widely used:

- *handshake*: Alice, Bob use their public-key certificates and private keys to authenticate each other and to exchange or create the shared secret
- *key derivation*: Alice, Bob use a shared secret to derive set of session keys
- *data transfer*: stream data transfer, data is seen as a series of records, not just one-time transactions
- *connection closure*: special messages are used to securely close the connection

Transport-layer security (TLS)

- TLS provides an Application Programming Interface (API) that any application can use
  - For example, TLS was added to HTTP:
    - Note: QUIC uses UDP instead of TCP



TLS: 1.3 cipher suite

The most recent version, TLS 1.3 (2018) has a more restricted set of options than TLS 1.2 (2008)

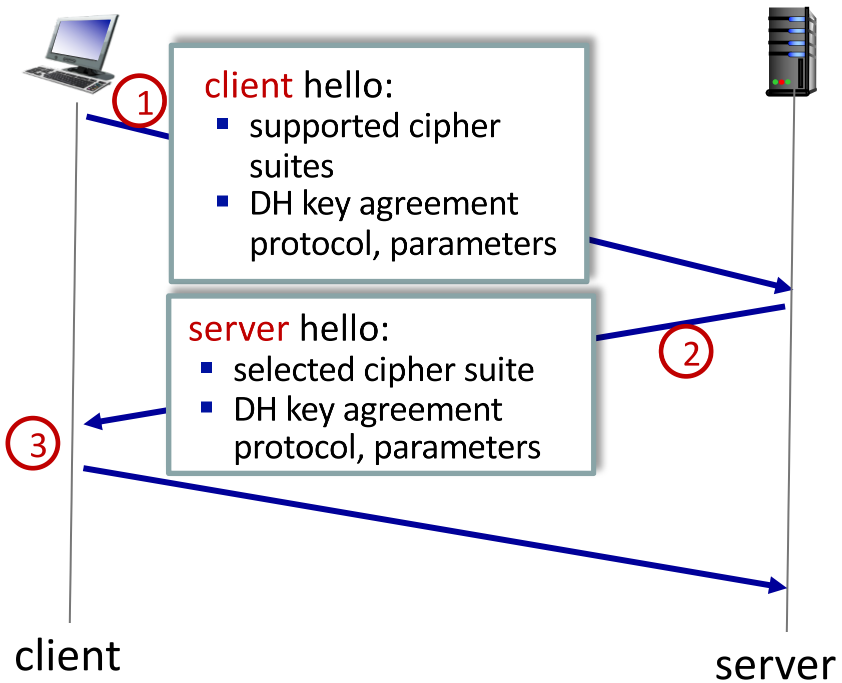
- Specifies a "*cipher suite*" of algorithms for key generation, encryption, MAC, digital signature
- Now *requires* the use of Diffie-Hellman (DH) for key exchange, rather than allowing DH or RSA
- Uses a set of algorithms that *combine* encryption and authentication
  - Employs *HMAC* (using either SHA 256 or 284) as the cryptographic hash function

TLS: 1.3 Encryption Keys

TLS uses a set of keys to exchange data

- Session keys are derived from a shared secret and possibly other data that will change over time (to avoid reuse of old session keys)
- To set up a connection, the client and server will exchange information about the cipher suite they will use and other protocol options
- Uses the Diffie-Hellman key exchange protocol to exchange keys between client and server
  - similar to the public/private key exchange approach we looked at

TLS 1.3 handshake:



1. client TLS hello msg:
   - *guesses* key agreement protocol, parameters
   - indicates cipher suites it supports
2. server TLS hello msg confirms:
   - key agreement protocol, parameters
   - cipher suite
   - server-signed certificate
3. client:
   - checks server certificate
   - generates key
   - can now make application request (e.g.., HTTPS GET)

HTTP vs HTTPS

- We looked at the HyperText Transfer Protocol earlier and noted that it does not encrypt the contents of the request and reply messages
- HTTPS is the solution to that issue and creates a secure *end-to-end* connection between client and server
  - HTTPS originally used the Secure Socket Layer (SSL) but now uses Transport Layer Security (TLS) at port 443

IP Security

- There are a broad range of application-specific security mechanisms for IP traffic
  - eg. S/MIME, PGP, Kerberos, SSL/HTTPS
- However, there are security concerns with other common Internet protocols
  - many protocols do not have security features
- Many IP applications could benefit from security implemented at the network level

Protocols Without Encryption

- HTTP
- FTP
- TFTP
- TELNET
- SMTP (email)
- NTP (time synchronization)
- SNMP (network management)
- and others...

IP-level Security (IPSec)

- Provide general IP Security mechanisms, including:
  - authentication
  - confidentiality
  - key management
- A study in 1994 identified this was needed
  - Goal: add authentication and encryption to IPv4
- Applicable for use over LANs, across public & private WANs, & for the Internet

IP Security Architecture

- Specification is quite complex with many standards groups involved in development:
  - Architecture
    - RFC4301 *Security Architecture for Internet Protocol*
  - Authentication Header (AH)
    - RFC4302 *IP Authentication Header*
  - Encapsulating Security Payload (ESP)
    - RFC4303 *IP Encapsulating Security Payload (ESP)*
  - Internet Key Exchange (IKE)
    - RFC4306 *Internet Key Exchange (IKEv2) Protocol*
  - But, uses existing cryptographic algorithms

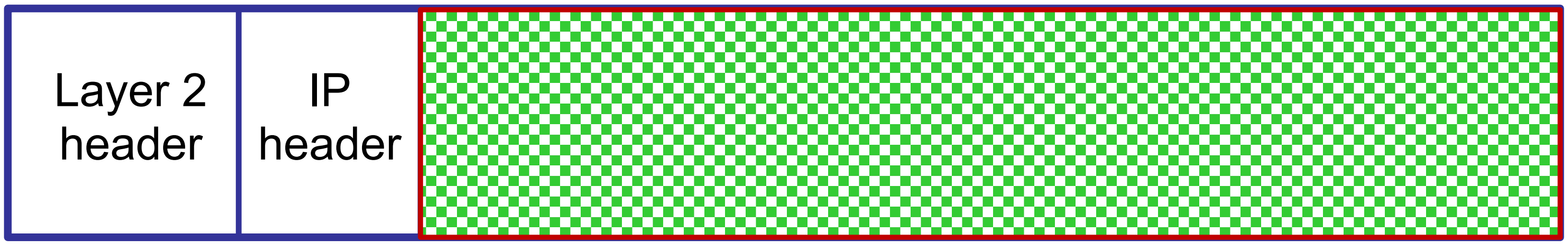
IPSec Services

- IPSec provides a range of services:
  - requires keys to control access to services
  - supports connectionless integrity (via encryption) and authentication of data's origin
  - supports rejection of replayed packets
    - to avoid man-in-the-middle & injection attacks
- IPSec includes two protocols that support secure communication services
  - Authentication Header (AH)
  - Encapsulating Security Payload (ESP)

IPSec – IP Security Protocol

- IP header can still be accessed, so it can run on existing routers/networks
- IP-level data (which can include TCP or other application protocols) is *encrypted*
  - provides end-to-end security between hosts
- Offers *authentication* of non-encrypted fields to detect modification in transit
  - For example, use SHA-1 to check integrity, AES or TripleDES for confidentiality

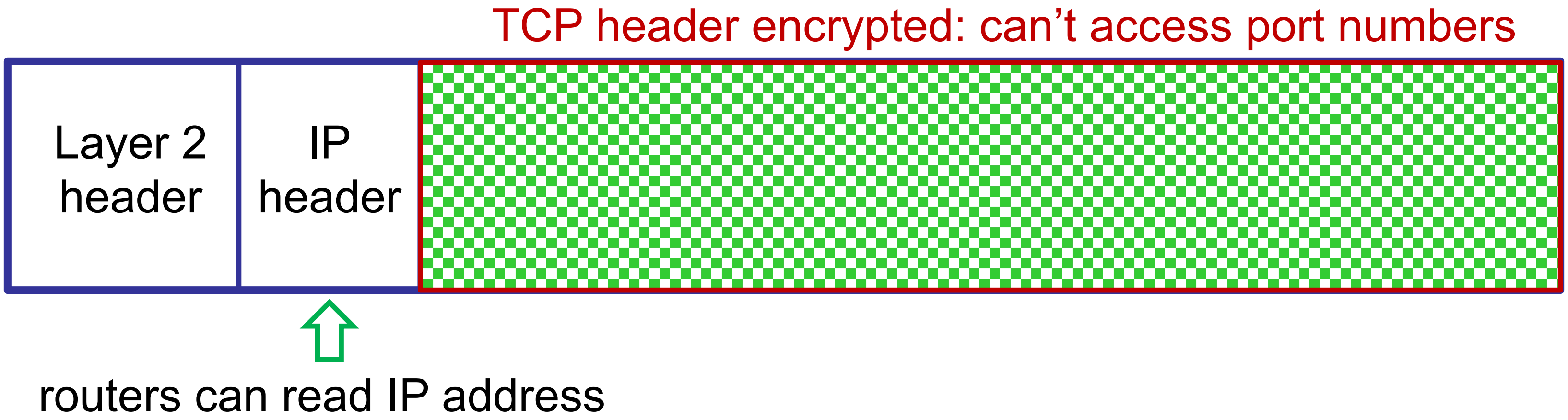
IPSec – IP Security Protocol



- Without encryption, all packet contents are *exposed* to monitoring and modification
- IPSec can *encrypt* the TCP header and data, leaving Layer 2 and IP headers open to support packet routing and delivery

IPSec Modes

- Transport Mode – *payload* is encrypted and hashed, IP header still exposed
  - works with unmodified routers
  - can't change port numbers – can't use NAT



IPSec Modes

- Tunnel Mode – *entire IP packet* encrypted and carried as data in an IPSec packet
  - creates a virtual private network
  - routers cannot see original packet headers
  - supports Network Address Translation (NAT)

original IP & TCP headers are inside encrypted section

routers can access/modify this *copy* of the IP address & port numbers, allowing port-addressable NAT

IPSec Services

- IPSec includes two protocols that support secure services
  - Encapsulating Security Payload (ESP)
    - provides encryption for *parts of* the packet
    - can't encrypt IP header fields in Transport mode, but can encrypt them in Tunnel mode
  - Authentication Header (AH)
    - provides *integrity check* for *entire packet*, including IP and TCP headers
    - Uses a hash function to detect changes

Encapsulating Security Payload (ESP)

- Provides message content confidentiality
- Provides limited traffic flow confidentiality
  - can hide original IP address and port number
  - can be combined with the Authentication Header for integrity checking
- Supports range of ciphers, modes, padding
  - Incl. DES, Triple-DES, RC5, IDEA, CAST, etc.
  - A variant of DES is most common
  - Pads data to fit blocksize & obscure traffic flow

Encapsulating Security Payload (ESP)

- ESP header is located after the IP header and before the Transport layer header or an encapsulating IP header (if Tunnel mode)
  - The ESP header is not encrypted, but the data that follows it will be encrypted
- ESP provides:
  - message content confidentiality
  - data origin authentication
  - connectionless integrity
  - an anti-replay service
  - limited traffic flow confidentiality

Authentication Header

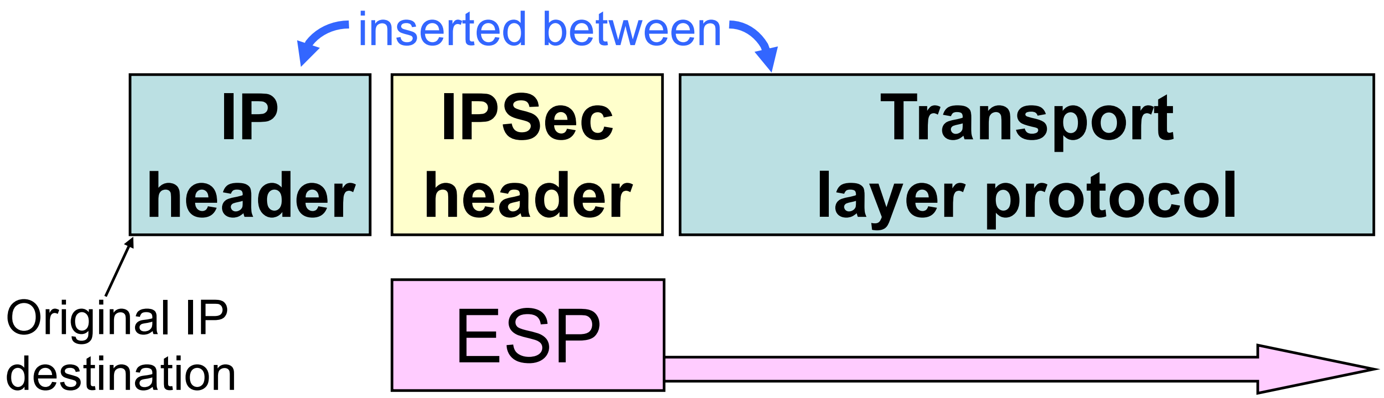
- Provides data integrity
  - Ensures that packet has not been tampered with
- Authentication protocol
  - Ensures that users can “trust” IP address source
  - Uses keyed MAC to authenticate
    - Symmetric encryption, e.g, DES
    - One-way hash functions with a shared secret key, e.g, HMAC-MD5-96 or HMAC-SHA-1-96
- Provides an anti-replay feature
- Includes an integrity check value (ICV)

Authentication Header

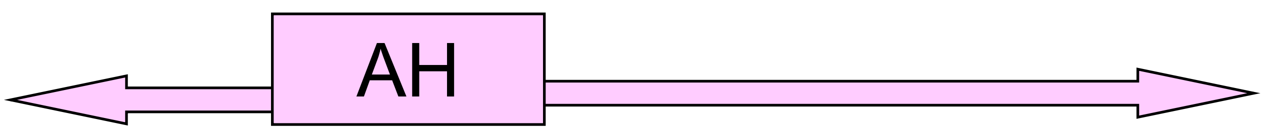
- Located behind the main IPv4 header to provide the three required IPSec parameters:
  - Security Parameters Index
  - Sequence Number
  - Authentication data
- Also contains the payload length and a pointer to the next header in the packet

1st byte	**2nd byte**	**3rd byte**	**4th byte**
Next Header	Payload Length	Reserved	
Security Parameters Index (SPI)			
Sequence Number Field			
Authentication data			

Transport Mode

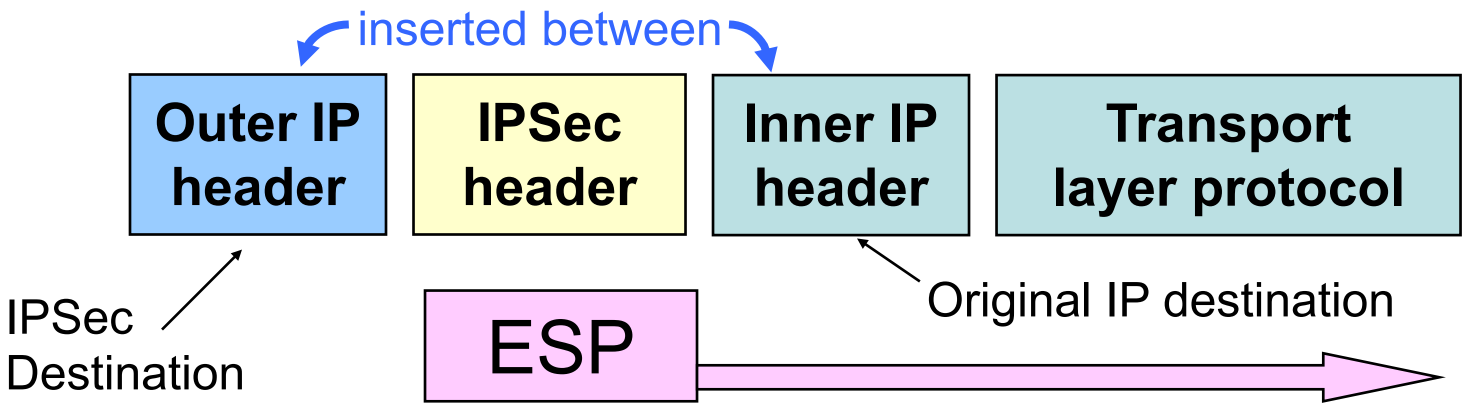


- ESP encrypts the transport-layer headers and the data in the original packet

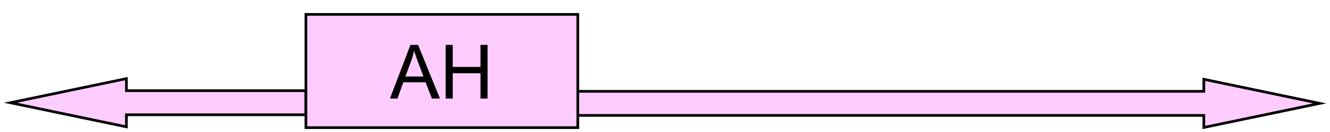


- AH provides an integrity check on the IP header as well as protecting the transport-layer headers and payload

Tunnel Mode



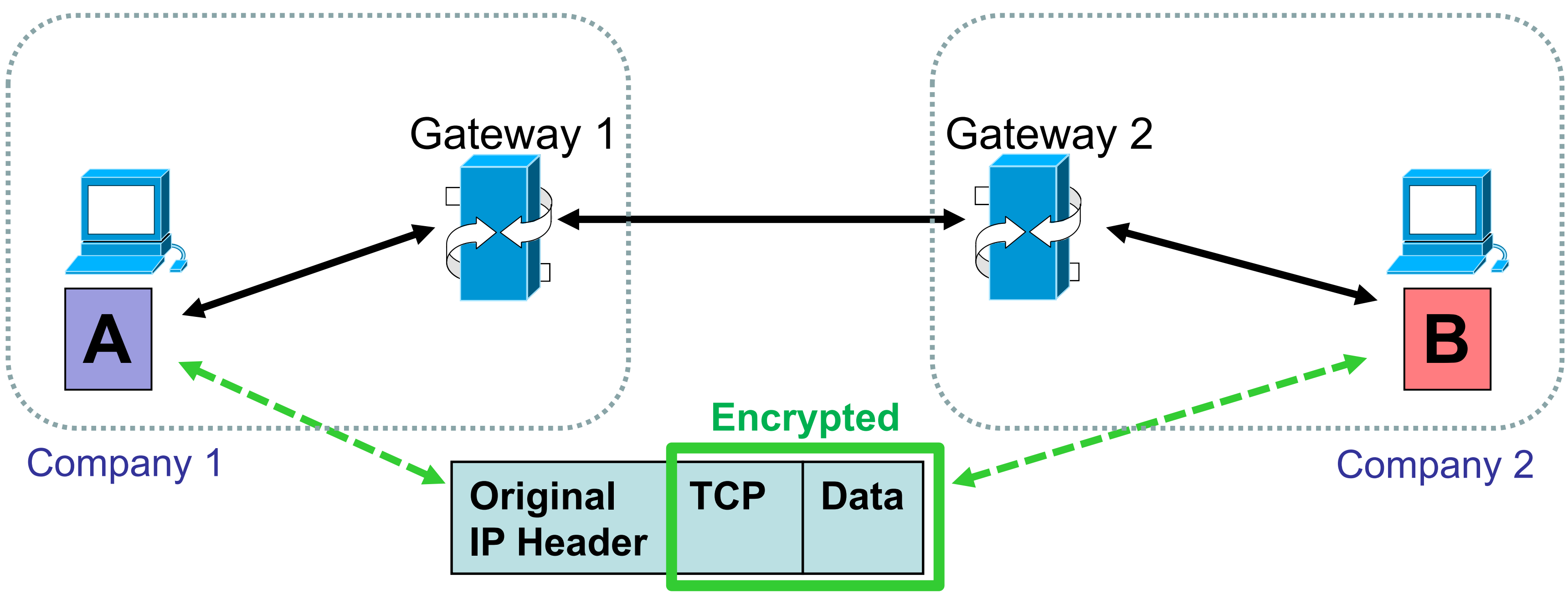
- ESP encrypts all of the headers and data in the original packet



- AH can be applied to both the inner packet and portions of the outer IP header

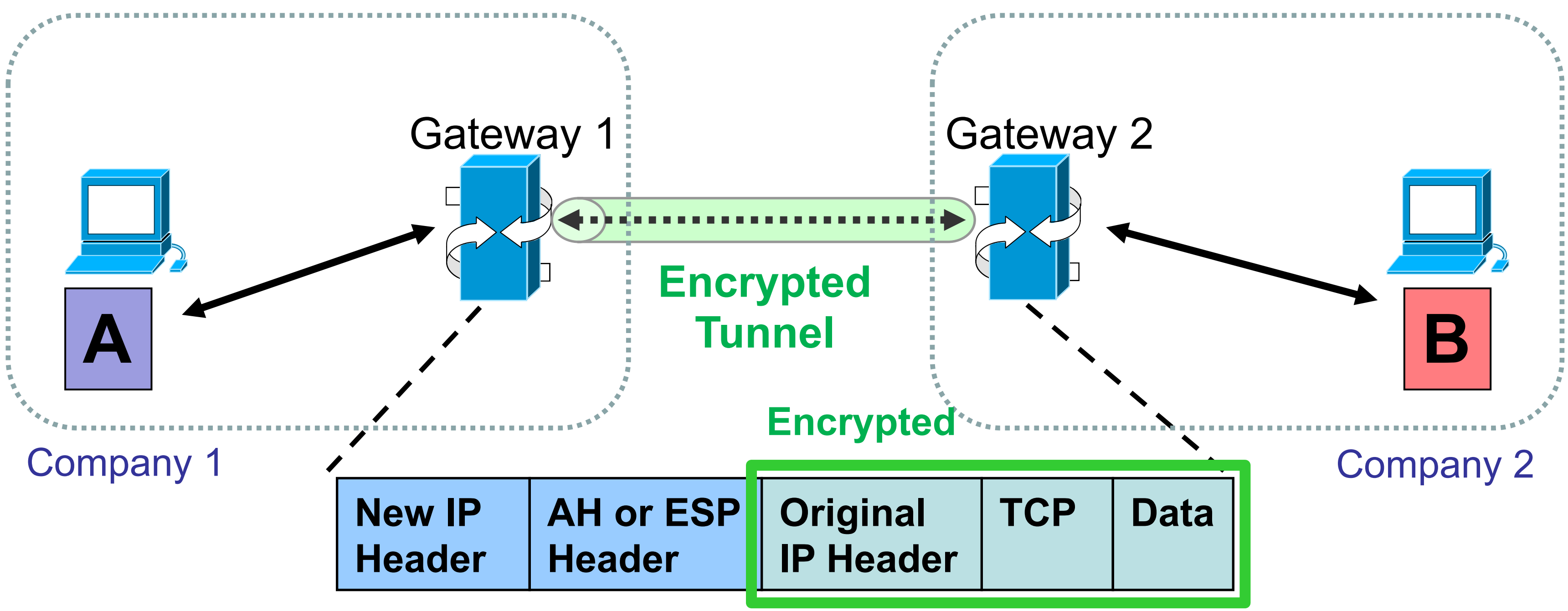
Transport Mode vs. Tunnel Mode

- Transport mode:
  - host ⇔ host



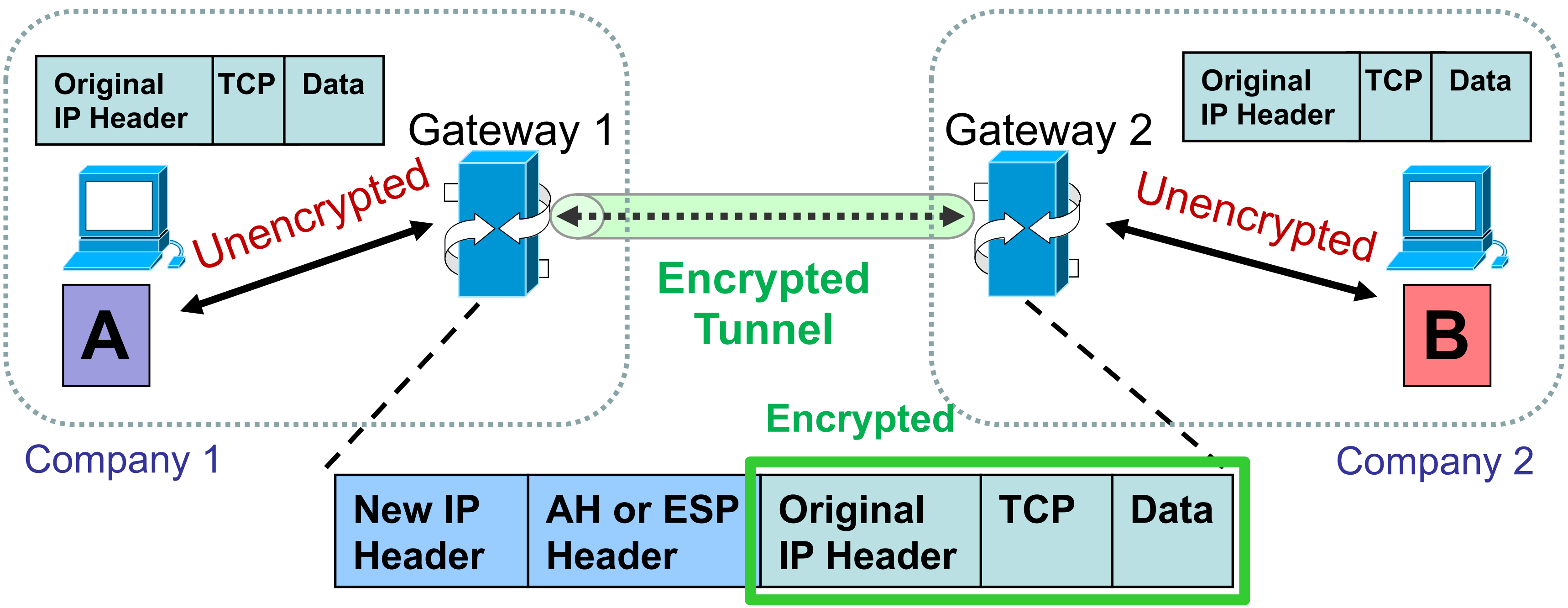
Transport Mode vs. Tunnel Mode

- Transport mode:
  - host ⇔ host
- Tunnel mode:
  - encrypted from gateway ⇔ gateway



Transport Mode vs. Tunnel Mode

- Transport mode:
  - host ⇔ host
- Tunnel mode:
  - encrypted from gateway ⇔ gateway



Transport Mode vs. Tunnel Mode

- Transport mode:
  - host ⇔ host
- Tunnel mode:
  - encrypted from gateway ⇔ gateway
  - encrypted from host ⇔ host or host ⇔ gateway