CS 419: Computer Security

Week 11: Protecting the Network
Secure Communication: VPNs

Paul Krzyzanowski

Fundamental Layer 2 & 3 Problems

- **No authentication**
  - Source: IP packets can be forged with fake source addresses
  - Destination: You don't know that the destination is legitimate
- **No integrity protection**
  - Data passes through untrusted hosts & can be modified in transit
- **No confidentiality**
  - Anyone along the route can see the traffic
- **Reliance on insecure protocols**
  - ARP, DHCP, BGP, DNS protocols – attacks can redirect traffic

Transport Layer Conversation Isolation:
Transport Layer Security (TLS)

We looked how TLS works previously – This is a review of what it gives us

Communication via an insecure network

Cryptography gives us the tools we need to communicate securely

Component	Goal	Example
Privacy	Make data unreadable without the key Encryption	AES, ChaCha20
Authentication	Validate the endpoints Public key cryptography	RSA, ECC, X.509 certificates
Integrity	Detect modifications MACs, Digital signatures, AEAD: Authenticated Encryption with Associated Data	HMAC, AES-GCM, ChaCha20-Poly1305
Key establishment	Securely agree on secret keys & provide perfect forward secrecy	Diffie-Hellman key exchange

Transport Layer Security

Goal: provide a *transport layer* security protocol

After setup, applications feel like they are using TCP sockets

SSL: Secure Socket Layer

Created with HTTP in mind

- Web sessions should be secure
  - Encrypted, tamperproof, resilient to man-in-the-middle attacks
- Mutual authentication is usually not needed
  - Client needs to identify the server, but the server isn't expected to know all clients
  - Rely on passwords or MFA to authenticate the client after the secure channel is set up

TLS: Transport Layer Security

Goal: provide a *transport layer* security protocol

After setup, applications feel like they are using TCP sockets

SSL: Secure Socket Layer – evolved to Transport Layer Security (TLS)

SSL evolved to TLS (Transport Layer Security)

SSL 3.0 was the last version of SSL … and is considered insecure

We now use TLS (but is often still called SSL)

Latest version = TLS 1.3 = SSL 3.4

The library is still called `libssl`, which is part of the OpenSSL distribution

TLS Goals

Provide authentication (usually one-way), privacy, & data integrity between two applications

Principles

- **Authentication** — *Client should be convinced it is talking with the correct server*
  - Use public key cryptography & **X.509 certificates** for authentication
  - Server side is always authenticated; client optional
- **Data confidentiality** — *Prevent eavesdropping*
  - Use **symmetric cryptography** to encrypt data
  - **Key exchange**: initial keys generated uniquely at the start of each session
- **Data integrity** — *Prevent tampering and man-in-the-middle attacks*
  - Include message integrity codes with transmitted data to ensure message integrity
  - Current versions use Authenticated Encryption with Associated Data (AEAD)
  - Earlier versions used HMAC
- **Perfect Forward Secrecy** — *Create keys for each session, so finding a past key won't decrypt future content*
  - Diffie-Hellman key exchange

TLS Protocol & Ciphers

Two sub-protocols

1. Handshake: authenticate & establish keys

- Authentication
  - X.509 certificates with RSA or Elliptic Curve Digital Signature Algorithm (or pre-shared key)
- Key exchange
  - Ephemeral Diffie-Hellman keys (keys generated for each session)

2. Record protocol: communication

- Data encryption options – *symmetric cryptography*
  - AES-128-GCM, AES-256-GCM, ChaCha20-Poly1305
- Data integrity – *message authentication codes*
  - AEAD – Authenticated Encryption with Additional Data – MAC based on selected encryption
  - HMAC-SHA256, HMAC-SHA384 (not needed if AEAD algorithms are used for encryption)

TLS 1.3 Basic Handshake

Goals:
1. Agree on a cipher suite
2. Establish trust
3. Agree on a master secret

Client

- “Hello”
- Diffie-Hellman public key
- Algorithms/modes

→ Server

Server

- “Hello”
- Diffie-Hellman public key
- Certificate
- (optional certificate request)
- Proof of private key possession

Client ←

Both sides now know what algorithms to use & have a D-H common key
Both parties send an HMAC using derived keys to confirm handshake integrity

Benefits & Downsides of TLS

Benefits

- Validates the authenticity of the server (if you trust the CA)
- Protects integrity of communications
- Protects the privacy of communications

Downsides

- Transport-layer solution: applications must explicitly use it
  - Only protects communication between two applications
- Longer latency for session setup (minimal with TLS 1.3)
- Just because a session is over TLS doesn't mean its trustworthy
  - Do you trust the remote side's certificate & that the server hasn't been hacked?

Client authentication Problem

- **TLS supports mutual authentication**
  - Clients can authenticate servers & servers can authenticate clients
- **Client authentication is almost never used**
  - Generating keys & obtaining certificates is not an easy process for users
  - Any site can request the user's certificate – *The user will be unaware their anonymity is lost*
  - Moving private keys around can be difficult
    - What about users on shared or public computers?
- **We usually rely on other authentication mechanisms**
  - Currently, most sites still use a username and password
  - … but there no danger of eavesdropping since the session is encrypted
  - Often use one-time passwords for two-factor authentication if worried about eavesdroppers at physical premises or credential theft (e.g., from the server or phishing attacks)

Securing Network Layer Communication:

Virtual Private Networks (VPNs)

Network vs. Transport Layer Secure Communication

TLS – Transport layer solution

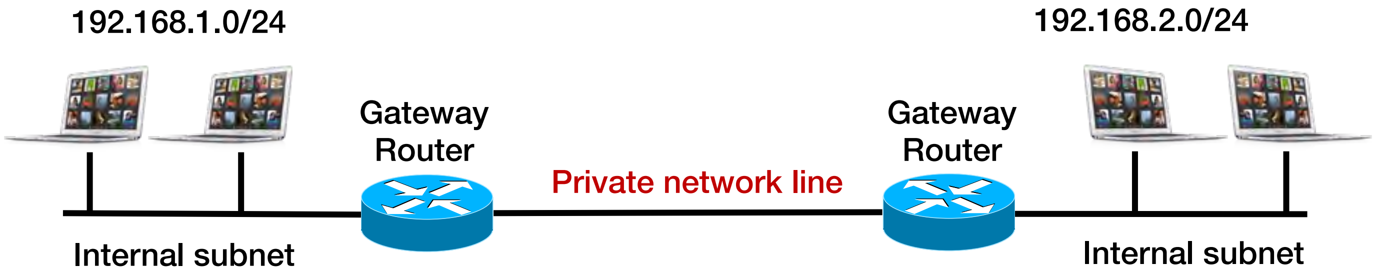
- It allows two applications to communicate via a secure channel
- The applications have to set up the connection

VPNs – Network layer solution

- Provide secure communication between networks
  or between a host and a network
- Establish a secure communication channel that can then be shared by all apps
- Applications are unaware: all communication across all applications is secure

Secure Communication: Private Networks

Connect multiple geographically-separated private subnetworks together



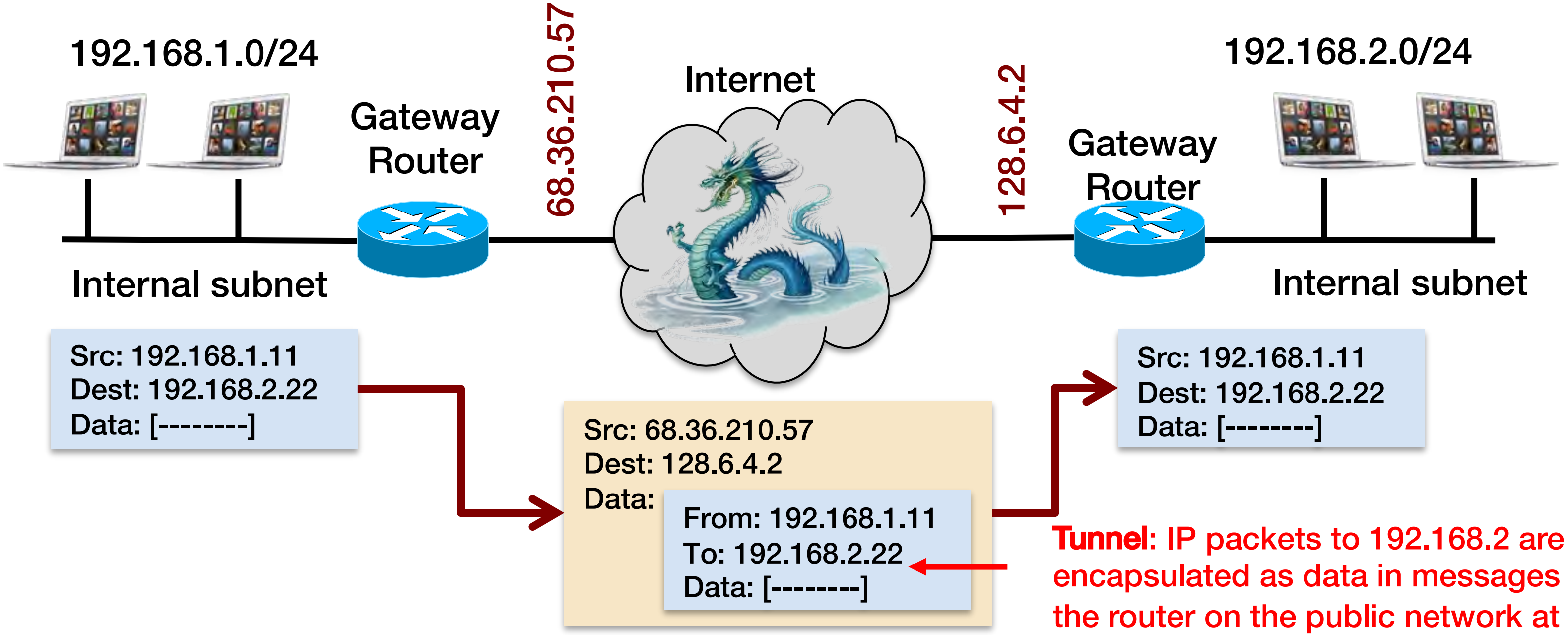
But this is expensive … and not feasible in most cases
(e.g., cost, bandwidth, use of cloud servers)

Example: companies like Amazon, Google, and Meta deployed private networks to connect their data centers

What’s a tunnel?

Tunnel = Packet encapsulation

Treat an entire IP datagram as payload on the public network



Tunnel: IP packets to 192.168.2 are encapsulated as data in messages to the router on the public network at 128.6.4.2

Virtual Private Networks

Take the concept of tunneling

… and safeguard the encapsulated data

VPN = tunnel + encryption + integrity + authentication

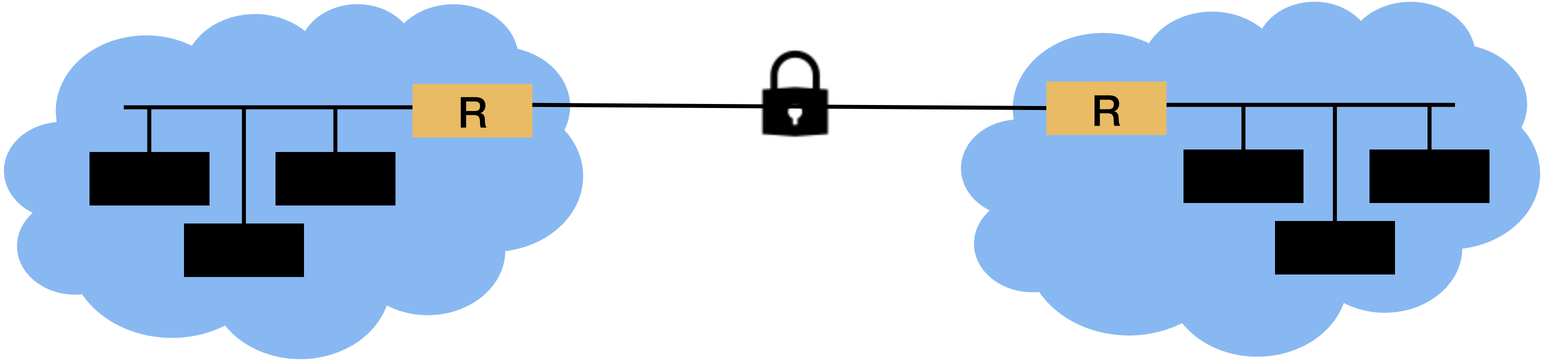
- **Add integrity (message authentication code)**
  - Ensure that outsiders don't modify the data
- **Encrypt the contents**
  - Ensure that outsiders can't read the data
- **Authenticate the endpoints**
  - Make sure you're connected to the right network/host

VPN Deployment Models

1. **Site-to-Site (Network-to-Network)**
   - Original model for VPNs
   - Connect geographically-separated networks
   - Two gateway routers establish a tunnel, making them appear as one.
2. **Remote Access (Host-to-Network)**
   - Support individual computers connecting to corporate network
   - The laptop appears to be part of the internal corporate network
   - Protects
3. **Remote Access (Host-to-Provider) – ExpressVPN, NordVPN, Proton**
   - Connect to provider – provider acts like gateway
   - Traffic appears to originate from the provider's service
   - Obscure geographic location

VPN Deployment Models: Evolving uses of VPNs

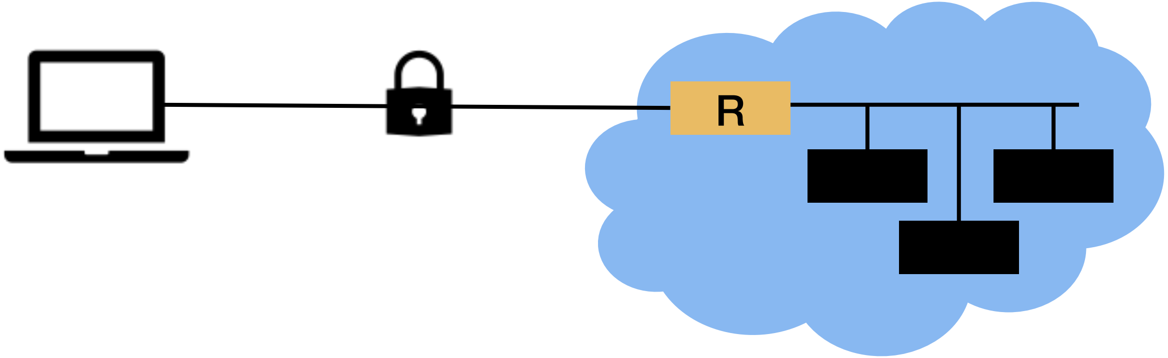
Network-to-Network



VPNs were first created to link geographically separated local area networks

VPN Deployment Models: Evolving uses of VPNs

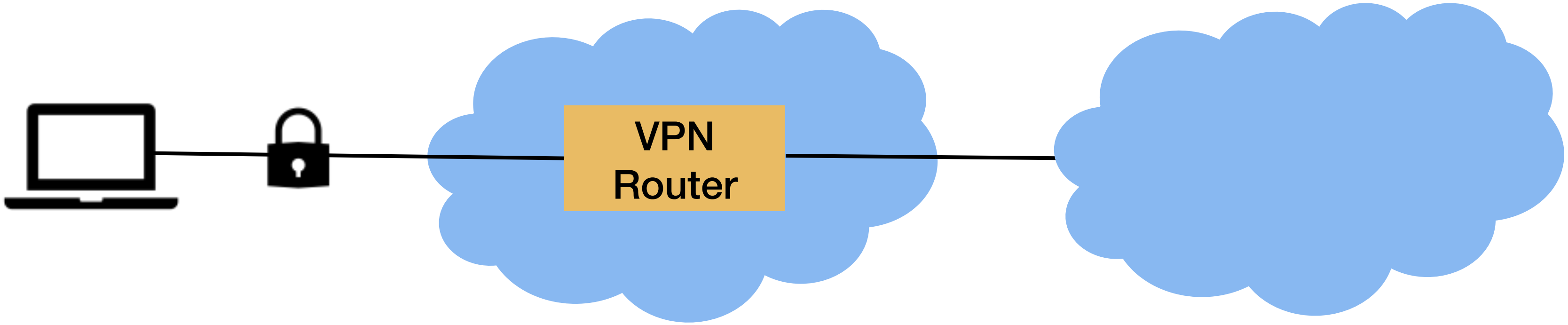
Host-to-Network



As broadband access and mobile work became common, VPNs enabled connecting a remote PC to a corporate network

VPN Deployment Models: Evolving uses of VPNs

Remote access to third-party provider



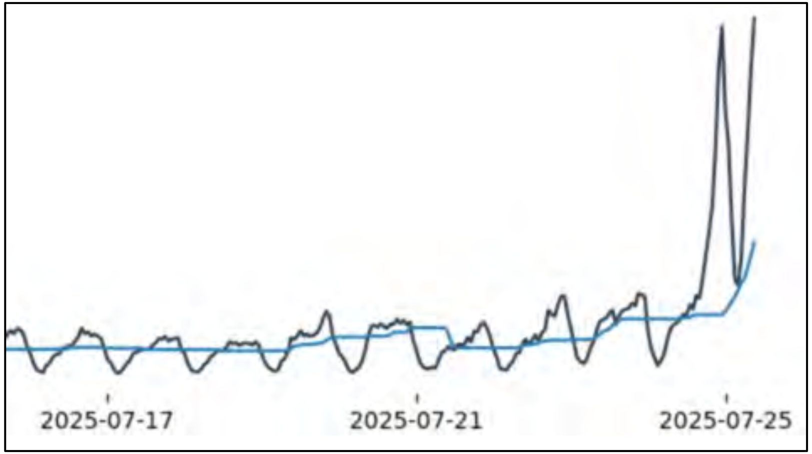
ExpressVPN, Proton VPN, Cyberghost, …

A common use now is to provide secure access to the VPN provider, which acts as a gateway – traffic appears to originate from the provider.

Evolving uses of VPNs

- **VPNs to third-party providers:**
  - Enable secure links from untrusted ISPs, such as public hotspots … but you must trust the VPN provider
  - Allow bypassing geographic restrictions (e.g., stream location-restricted video)

ProtonVPN saw a 1400% hourly increase in VPN activity immediately after the UK began enforcing age verification rules for explicit content



https://protonvpn.com/internet-censorship-observatory

Virtual Private Networks

There are lots of VPN implementations

We'll look at just three popular ones

1. **IPsec – the first standard (1990s)**
   – Implemented in the kernel at the network layer
   – Standardized, widely deployed, complex
2. **OpenVPN**
   – Runs in user space leveraging TLS
   – Highly portable across nearly all platforms
3. **WireGuard**
   – Runs in kernel space but communicates via the transport layer (UDP)
   – High speed, low overhead, formally verified

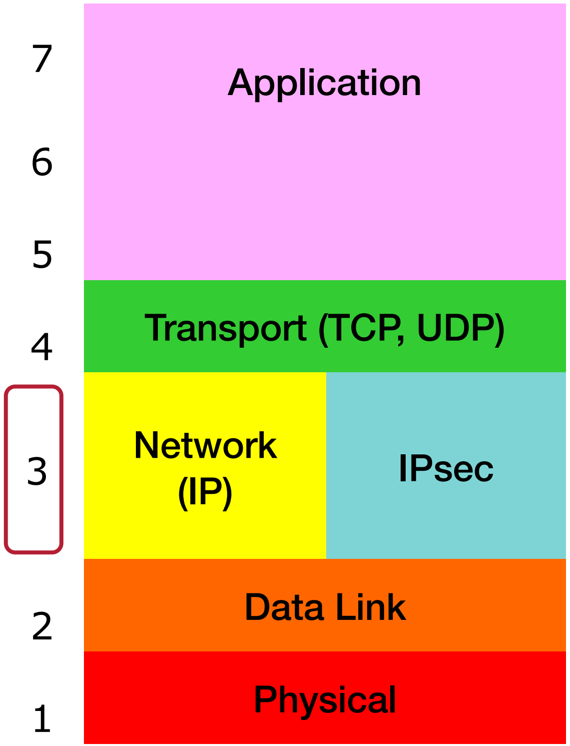
IPsec (implemented in kernel at network layer)

Internet Protocol Security

End-to-end security at the IP layer

Two protocols:

- **IP Authentication Header** Protocol (AH)
  - Authentication & integrity of payload and header
  - *Provides authentication & integrity*
- **Encapsulating Security Payload** (ESP)
  - AH features + encryption of payload
  - *Adds confidentiality*



IPsec is a separate protocol from UDP or TCP – protocols 50 (ESP) & 51 (AH) in the IP header.
Layer 3 protocol – gateway routers are responsible for encapsulating/decapsulating

Tunnel mode vs. transport mode

IPsec Tunnel mode

- Communication between gateways: *network-to-network* or *host-to-network*
- Entire IP datagram is encapsulated
  - The system sends IP packets to various addresses on subnet
  - A router (tunnel endpoint) on the remote side extracts the datagram and routes it on the internal network
  - The outer IP packet (tunnel) sends data to the gateway router

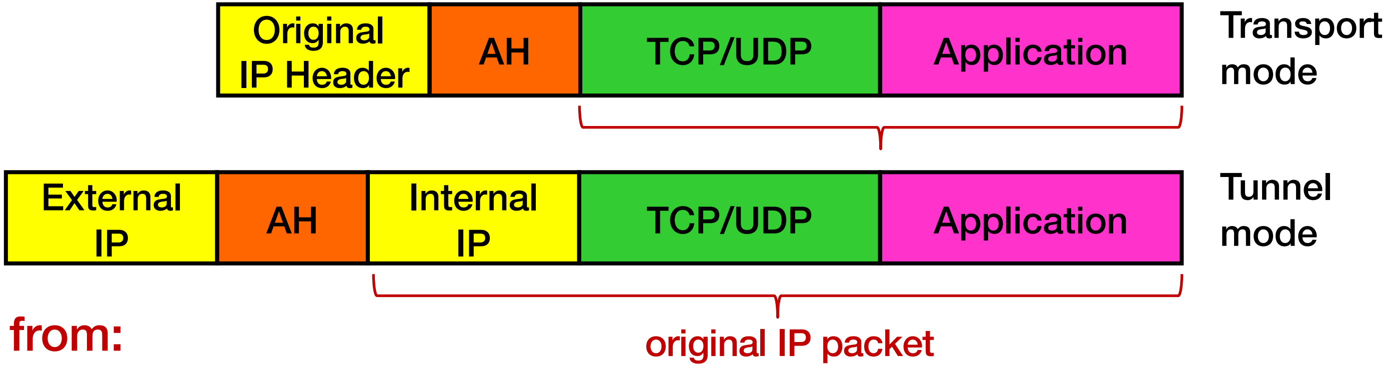
IPsec Transport mode (Note: not transport layer!)

- Communication between hosts
- IP header is not modified but payload is protected
  - The system communicates directly with only one other system
  - The original IP header is needed because the packet is sent directly to that host

IPsec Authentication Header (AH)

Guarantees integrity & authenticity of IP packets

- MAC for the contents of the entire IP packet
- Computed over unchangeable IP datagram fields (e.g., not TTL or fragmentation fields)



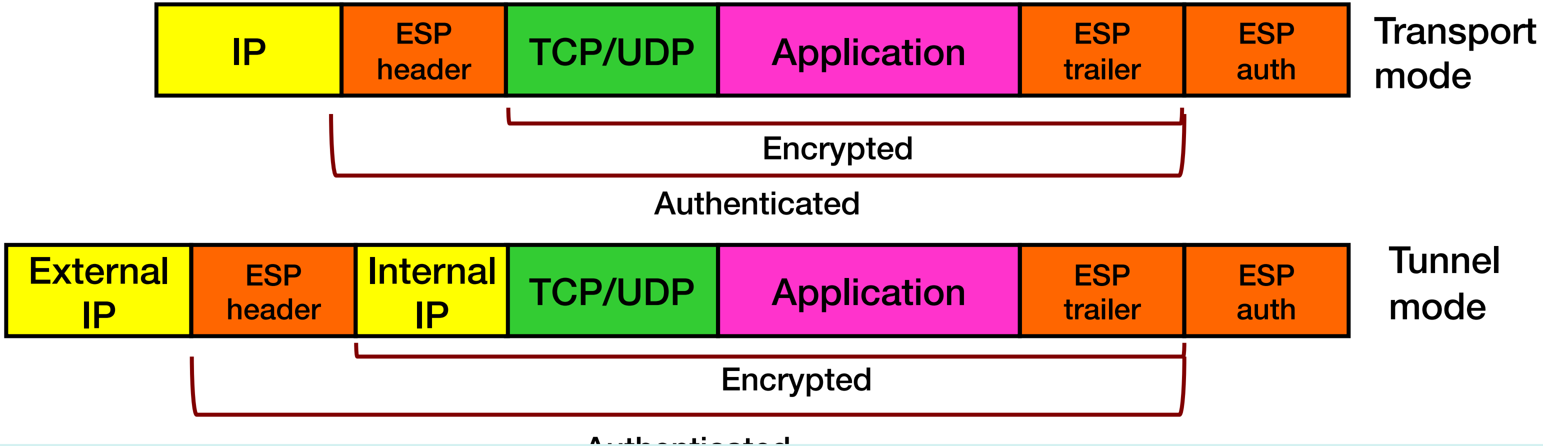
Protects from:

- Tampering
- Forging addresses
- Replay attacks (sequence number in MAC-protected AH)

IPsec Encapsulating Security Payload (ESP)

Encrypts the entire payload

- And includes authentication of payload and IP header (everything AH does) (this may be optionally disabled – but you don't want to)



Why do we have AH and ESP, when ESP does everything AH does and more?

IPsec is the oldest VPN protocol, dating to the 1990s, when encryption consumed a significant % of CPU time, so using AH provided integrity without the cost of encryption. There's no good reason to use AH today.

IPsec algorithms

- **Authentication: Certificates or pre-shared key authentication**
  - Public keys in certificates (RSA or ECC) used for authenticating users (authenticate by using your private key to decrypt data that was encrypted with the public key in your certificate)
  - Pre-shared keys = authenticate via a shared key that was set up ahead of time
- **Key exchange – Diffie-Hellman**
  - Diffie-Hellman to create a common key for key generation
  - Key lifetimes determine when new keys are regenerated
  - Random key generation ensures Forward Secrecy
- **Confidentiality – symmetric algorithm**
  - 3DES-CBC, AES-CBC, AES-CTR, …
- **Integrity protection & authenticity – MACs**
  - HMAC-SHA1, HMAC-SHA2

IPsec

Advantages

- Standard
- Widely deployed
- Efficient, kernel-level
- Transparent to apps

Disadvantages

- Complex implementation
- Complex configuration
- Network layer leads to problems with some NAT gateways

OpenVPN

1st open-source VPN protocol

Step 1: Tunnel setup

- OpenVPN software runs in user space: creates tunnels over TCP or UDP
- A virtual network interface is created to intercept traffic for the VPN
  - Clients can get unique IP addresses
  - Most operating systems provide a TUN (network TUNnel) interface that allows passing IP packets from the kernel to a user process

OpenVPN

Step 2: Key exchange & authentication (Control channel)

- Supports TLS for key exchange and authentication (not transport)

- **Two authentication modes**
  1. Pre-shared static keys
     - Four independent keys: HMAC send, HMAC receive, encrypt, decrypt
  2. TLS + certificates (most common)
     - Bidirectional authentication: both sides present a certificate
     - Send a list of supported ciphers

- Diffie-Hellman is used to establish a shared session key

TLS Control channel:

- **Initial TLS handshake**
- **Kept active for the session**
- **Periodic renegotiation of session keys**
- **Keep-alive messages**
- **Termination messages**

OpenVPN

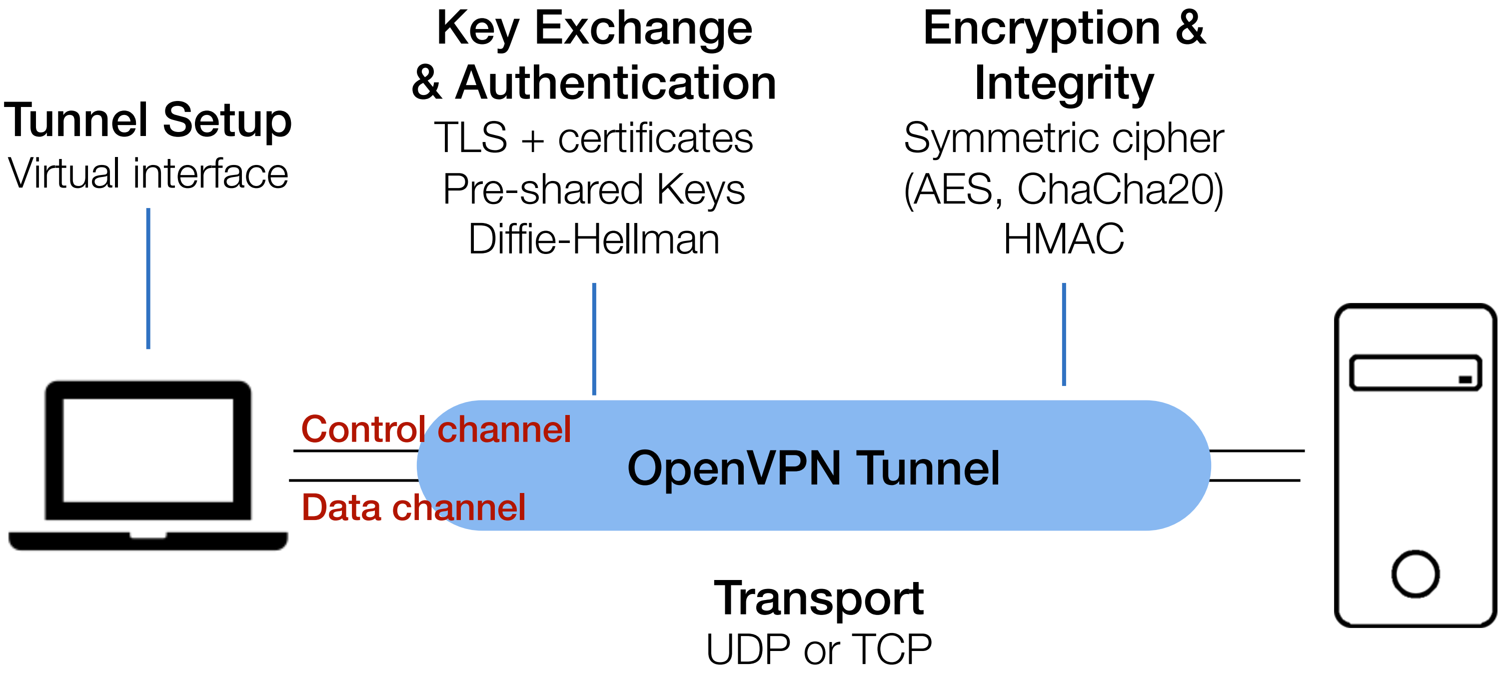
Step 3: Data encryption & Integrity

- Symmetric encryption: common algorithms are AES, ChaCha20
- HMAC for integrity: commonly HMAC-SHA256
- Forward secrecy achieved if using ephemeral keys (non-pre-shared)

Transport options

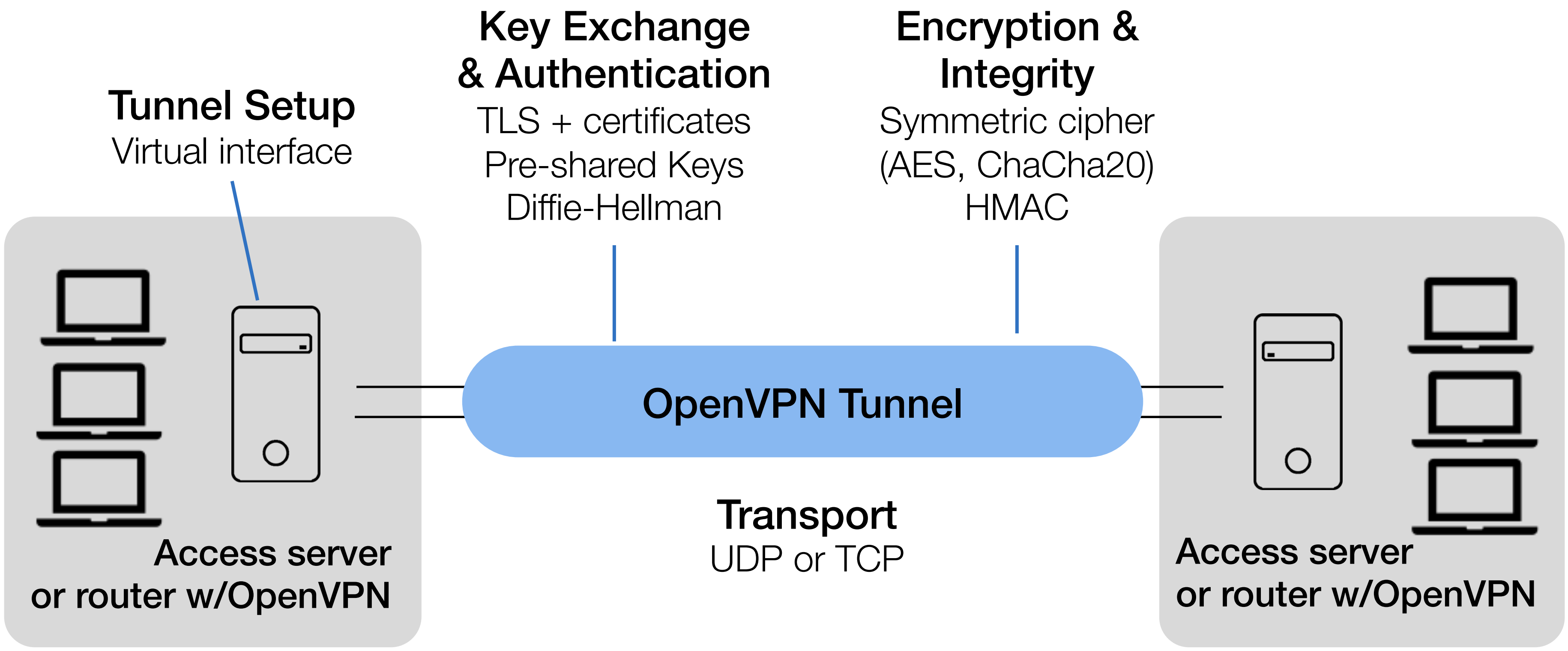
- OpenVPN can run over TCP or UDP
  - UDP: great for performance
  - TCP: great for bypassing firewalls

OpenVPN



Tunnel Setup
Virtual interface
Key Exchange
& Authentication
TLS + certificates
Pre-shared Keys
Diffie-Hellman
Encryption &
Integrity
Symmetric cipher
(AES, ChaCha20)
HMAC
Control channel
Data channel
OpenVPN Tunnel
Transport
UDP or TCP

OpenVPN – Site-to-Site Communication



OpenVPN

Advantages

- Open source
- Runs in user space
- Standard TLS protocol
- Highly portable

Disadvantages

- Slower because of user-space implementation

WireGuard

Goal: simple design – focus on latest algorithms & high performance

- Code has been formally validated: the codebase is only 4,000 lines of code

Communication & tunneling via UDP messages

- **Setup**
  - Hosts share **public keys** with each other – *hosts are identified by their public keys (no certificates)*
  - Keys are associated with IP addresses that should be sent via the tunnel
- **Communication initialization (handshake)**
  - **Diffie-Hellman key exchange** to establish shared keys (Elliptic curve algorithm)
  - Re-established every minute to create new keys – no protocol negotiation
- **Data transmission of packets**
  - Encryption: **ChaCha2** stream cipher
  - Message Authentication Code: **Poly1305** *hash*(message, secret)

WireGuard

Advantages

- Open source
- Super efficient kernel space implementation
- Great for embedded systems
- No protocol negotiation for ciphers
- Easy to configure

Disadvantages

- None really
- Hasn’t been around long but has been verified formally
- No/limited support in many VPN gateway products

The End