Chapter 5

Secure Channels (Communications Security)

Remember!

Secure channel

- A secure channel is
  - a communication channel between two people/services/objects (principals)
  - both are mutually authenticated
  - channel is encrypted and its integrity is secured against eavesdropping / modification (add/delete/change) / generation (from nothing)
  - intention is to force attackers (including telco/NSA level) from passive into active attacks, because passive attacks are not detectable and active attacks are far more costly
- Basic requirements
  - standard security requirements!
    - (mutual) authentication
    - confidentiality
    - integrity protection
  - further requirements strongly dependent on user / application
  - combination of methods to fulfill all requirements

Different from CIA triad for systems

Secure channel requirements

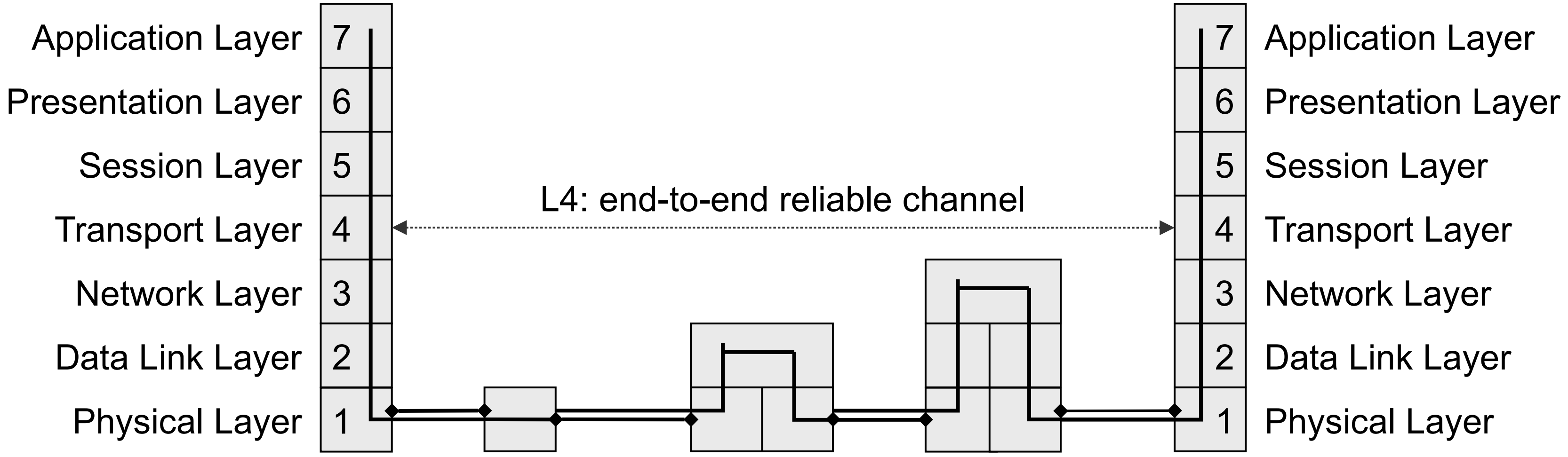
- Initial key exchange
  - when key exchange is insecure, then all following cryptographic methods are useless!
  - in most protocols, this is the weakest part
  - options:
    - “in-band”: DH + authentication of key
    - “out-of-band”: exchange over other channel
- Management of session keys
  - hybrid crypto systems for better performance ⇒ session key (symmetric) can be different from initial key (asymmetric)
  - should be changed/updated regularly to counter statistical attacks
    - e.g. for each message
    - or after X messages, after Y seconds, after Z bytes, etc.

Secure channel requirements

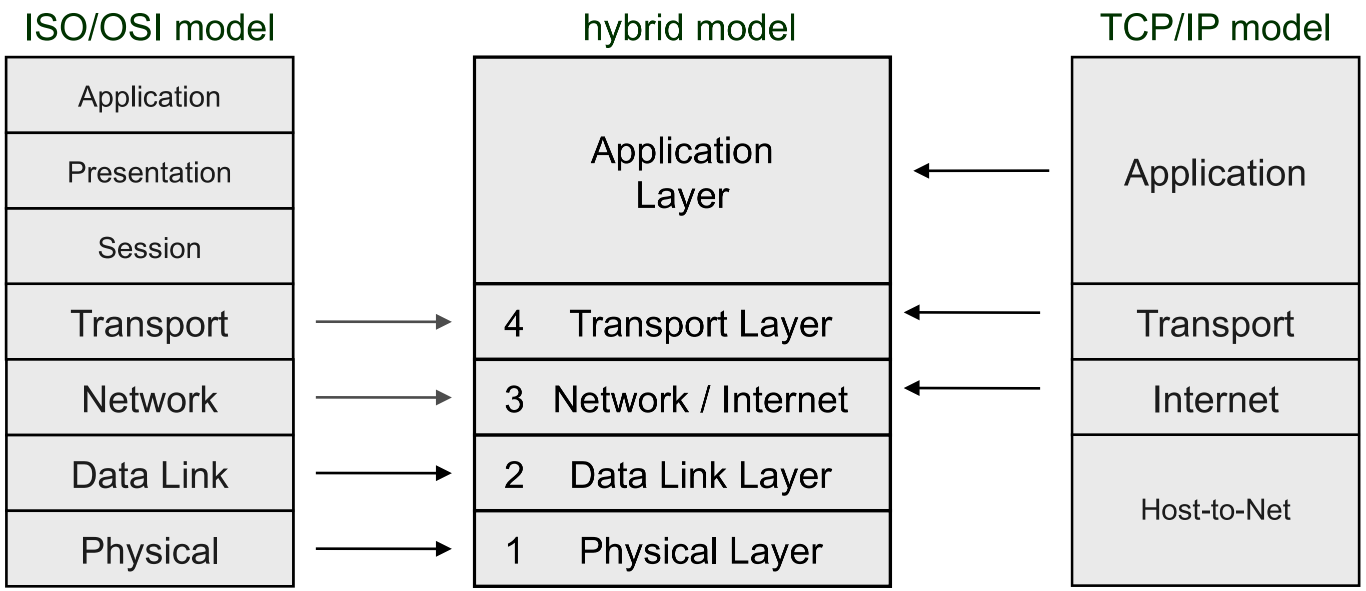
- Exchanging crypto algorithms
  - old algorithms might become insecure
    - cryptanalysis
    - faster hardware
  - regulations on algorithms use (country-specific, enterprise policies, etc.)

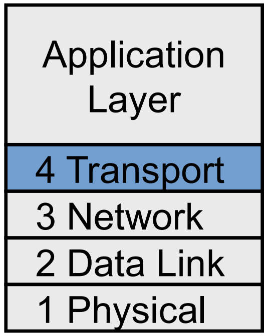
    ⇒ must be possible to exchange algorithms without modifying the protocol
- Further requirements
  - sequence numbers to counter replay/suppression/reordering attacks
  - time stamps to counter delay attacks
  - randomization to counter statistical cryptanalysis
  - compression
    - impossible after correct encryption
    - thus, compress before encryption in the secure channel protocol

Secure channel layers



Secure channel layers

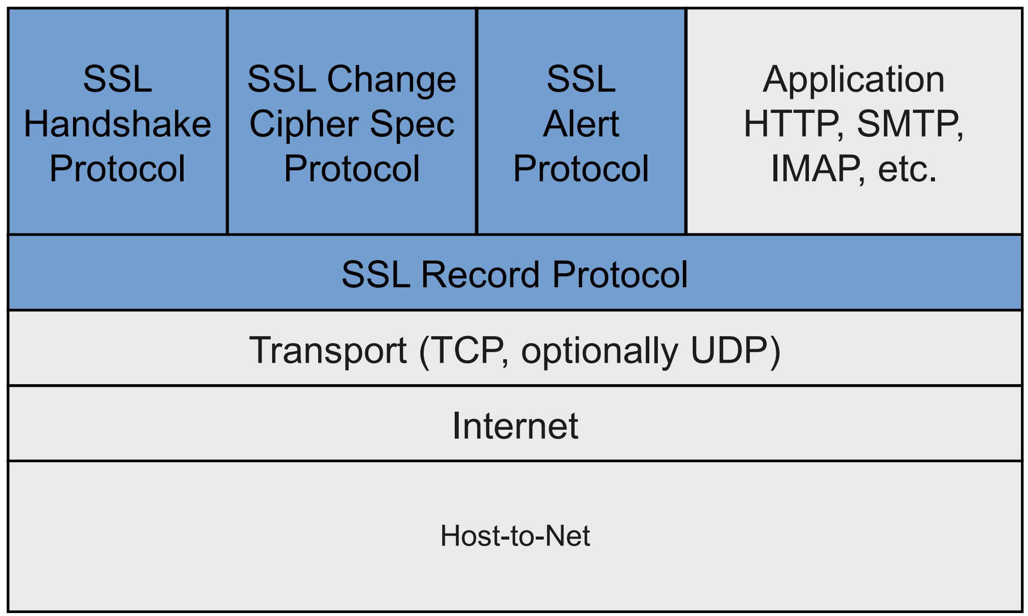




Secure Socket Layer (SSL)

- Originally developed by Netscape
- Version 3 designed with public input
- Subsequently became Internet standard known as **TLS (Transport Layer Security)**
- Normally uses TCP to provide a reliable end-to-end service (but can be run on top of UDP in special cases)
- SSL has two layers of protocols

SSL/TLS architecture



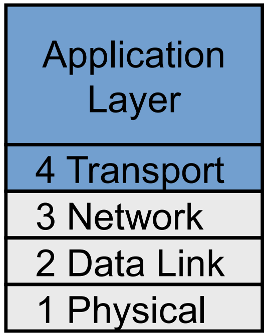
Application Layer
4 Transport
3 Network
2 Data Link
1 Physical

Transport Layer Security (TLS)

- TLS 1.0: IETF standard RFC 2246 similar to SSLv3
- With minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
    - ➔ based on HMAC using SHA-1 or MD5
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate types and negotiations
  - changes in crypto computations and padding
- Since then important improvements in TLS 1.1, 1.2, and recently 1.3
  - Why “important”? Security problems were discovered!

TLS 1.3

- Published in final standard form in August 2018 as RFC 8446
- Faster (but with security drawbacks when server is compromised)
  - 0-RTT (zero round trip time) startup reduces one roundtrip in establishing TLS handshake and caches result for next session
- More secure
  - removes some features and crypto suites:
    - SHA-1, RC4, DES, 3DES, MD5 primitives
    - CBC mode
    - RSA key exchange (see padding oracle attacks)
    - non-ephemeral Diffie-Hellman groups (see CVE-2016-0701)
    - EXPORT strength ciphers (see FREAK and LogJam)
  - enforces Forward Secrecy (FS)
- For details, see standard
  - or e.g., https://tls13.ulfheim.net/



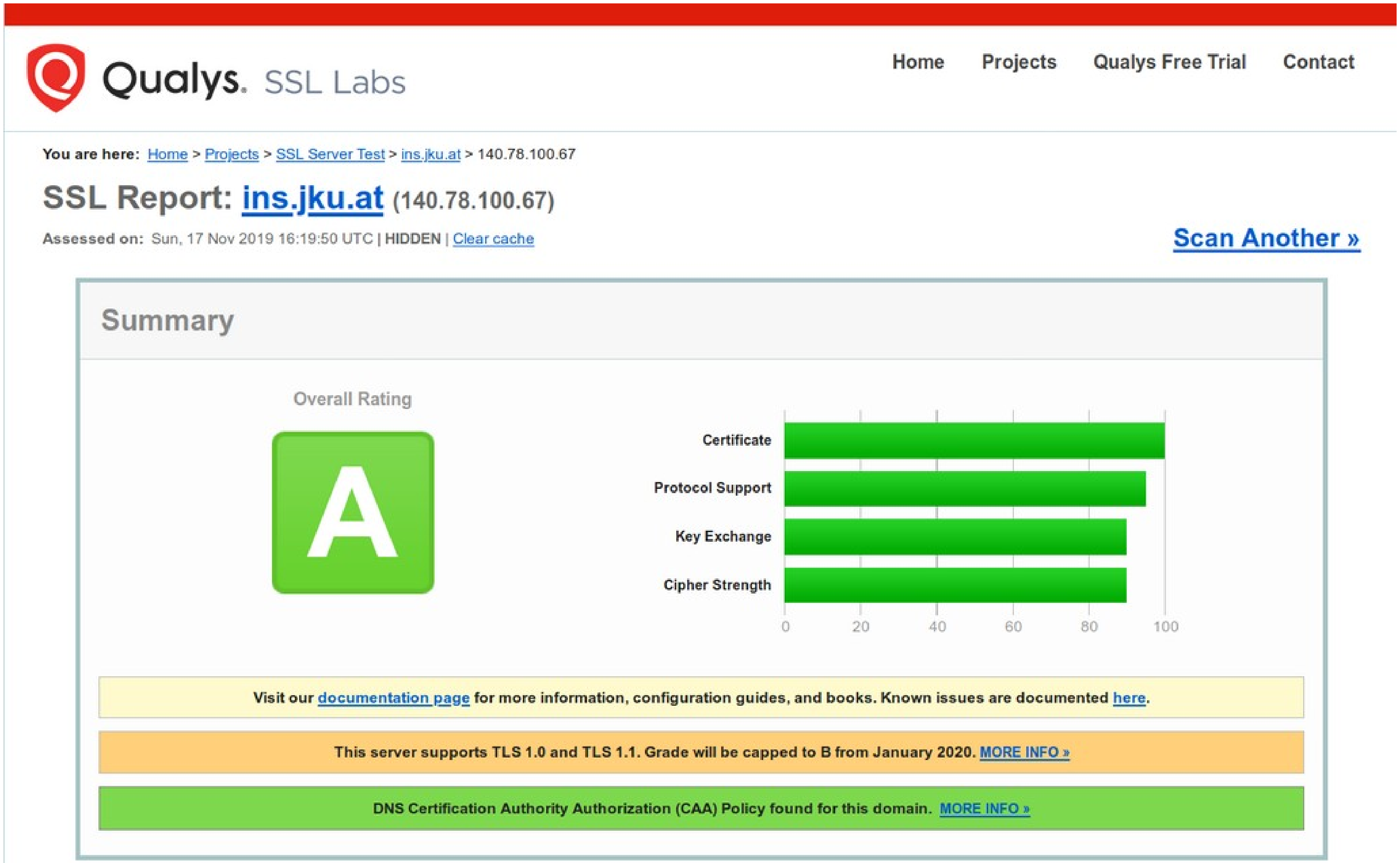
HTTPS

- HTTPS (HTTP over SSL)
  - combination of HTTP and SSL/TLS to secure communications between browser and server
    - ➔ documented in RFC2818
    - ➔ no fundamental change using either SSL or TLS
- Use `https://` URL rather than `http://`
  - and port 443 rather than 80
- Encrypts
  - URL, document contents, form data, cookies, HTTP headers
- Does **not** encrypt
  - IP address of server, IP address of client: **Network** layer
  - hostname (virtual hosting: multiple domain names on a single server)
    - Which certificate should the server present if it does not yet know which one the client would like to access?
    - TLS 1.3 allows “encrypted SNI” / “encrypted ClientHello” to solve this issue

TLS security issues

- http://bristolcrypto.blogspot.co.at/2013/08/why-does-web-still-run-on-rc4.html
- https://wiki.thc.org/ssl
- Recent attacks on TLS:
  - CRIME → compression in TLS/SSL problematic
  - BEAST → CBC usage problematic → either don't use CBC or switch to TLS 1.2
  - Lucky-13
  - RC4 problems (http://www.isg.rhul.ac.uk/tls/) → don't use RC4
- Current recommendation for TLS clients and servers
  - enable TLS >=1.2, best 1.3 (most important!)
  - switch to secure cipher suites, recommended AES-GCM or AES-CCM
  - **enable perfect forward secrecy (PFS)**, for performance reasons probably ECDHE
- Test clients and servers at https://www.ssllabs.com

SSL Server Test

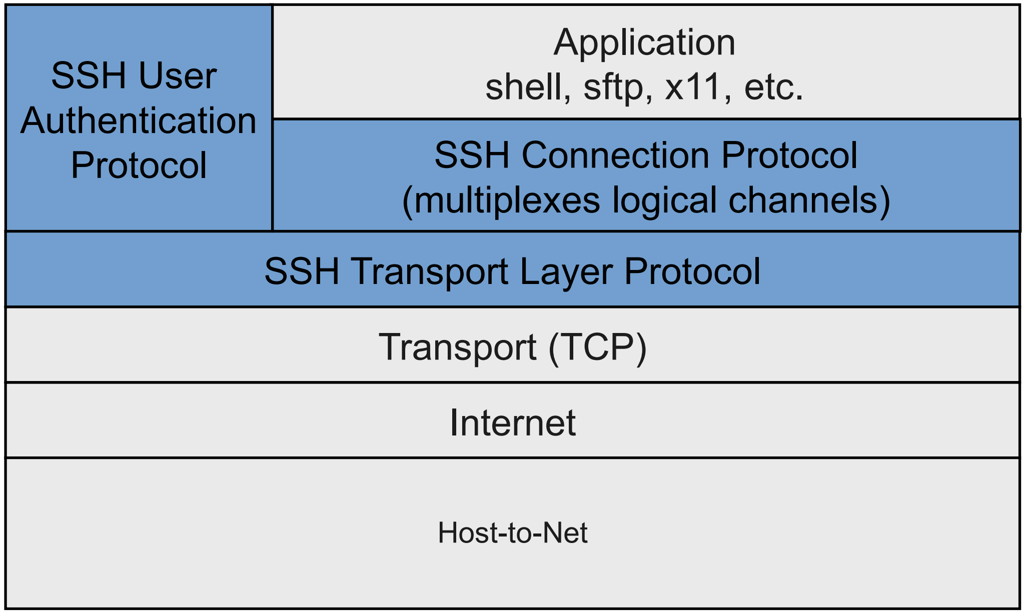


Application Layer
4 Transport
3 Network
2 Data Link
1 Physical

Secure Shell (SSH)

- Protocol for secure network communications
  - designed to be simple and inexpensive
- SSH1 provided secure remote logon facility
  - replace TELNET and other insecure schemes
  - also has more general client/server capability
- SSH2 fixes a number of security flaws
- Documented in RFCs 4250 through 4254
- SSH clients and servers are widely available
- Method of choice for remote login / X tunnels

SSH protocol stack



SSH transport layer protocol

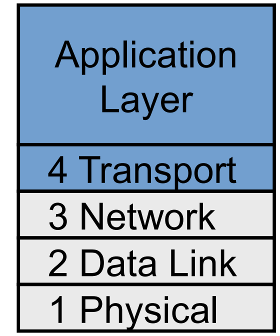
- Server authentication occurs at transport layer, based on server/host key pair(s)
  - server authentication requires clients to know host keys in advance
- Packet exchange
  - establish TCP connection
  - can then exchange data
    - ➔ identification string exchange, algorithm negotiation, key exchange, end of key exchange, service request
  - using specified packet format

SSH user authentication protocol

- Authenticates client to server
- Three message types:
  - SSH_MSG_USERAUTH_REQUEST
  - SSH_MSG_USERAUTH_FAILURE
  - SSH_MSG_USERAUTH_SUCCESS
- Authentication methods used
  - public-key, password, host-based

SSH connection protocol

- Runs on SSH Transport Layer Protocol
- Assumes secure authentication connection
- Used for multiple logical channels
  - SSH communications use separate channels
  - either side can open with unique id number
  - flow controlled
  - have three stages:
    - opening a channel
    - data transfer
    - closing a channel
  - four types
    - session: remote program execution, typically a shell
    - X11: forwarding mouse/keyboard and screen (remote desktop)
    - forwarded-tcpip: connections to remote computer should be sent to local one
    - direct-tcpip: connection to local computer is sent out from remote one

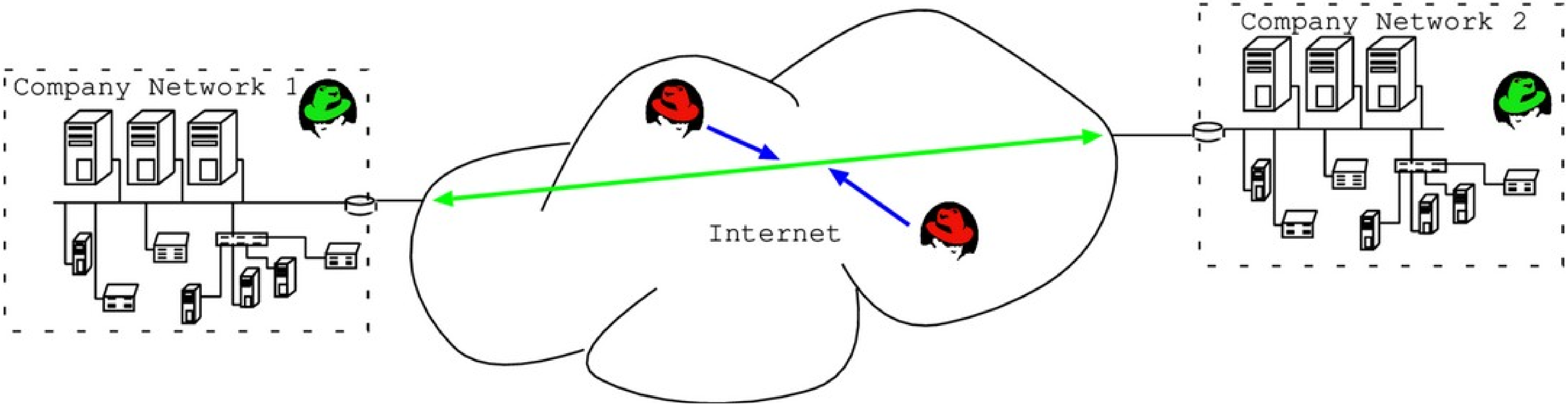


Port forwarding

- Convert insecure TCP connection into a secure SSH connection
  - SSH Transport Layer Protocol establishes a TCP connection between SSH client and server
  - client traffic redirected to local SSH, travels via tunnel, then remote SSH delivers to server
- Supports two types of port forwarding
  - local forwarding – SSH **client** acts as TCP server, traffic to that port is forwarded through SSH tunnel and SSH server connects as client to specific target server
    - “forwards” TCP tunneling
  - remote forwarding – SSH **server** acts as TCP server, traffic to that port (on the server) is forwarded through SSH tunnel and SSH client connects to specific target server
    - “backwards” TCP tunneling

Virtual Private Networks (VPNs)

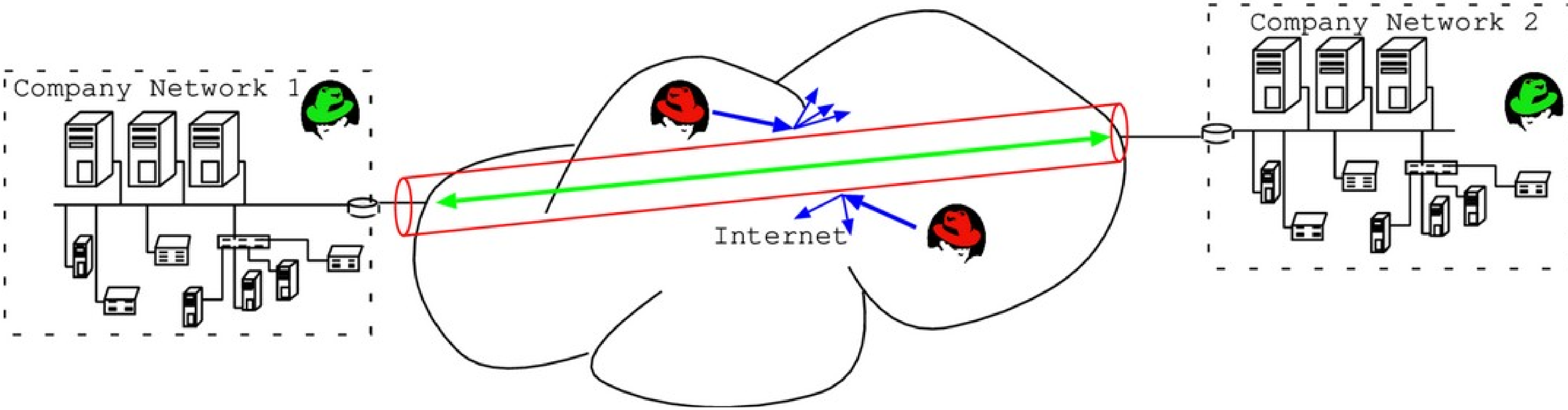
Application Layer
4 Transport
3 Network
2 Data Link
1 Physical



Acknowledgments: diagram by Utz Roedig at Lancaster University

Application Layer
4 Transport
3 Network
2 Data Link
1 Physical

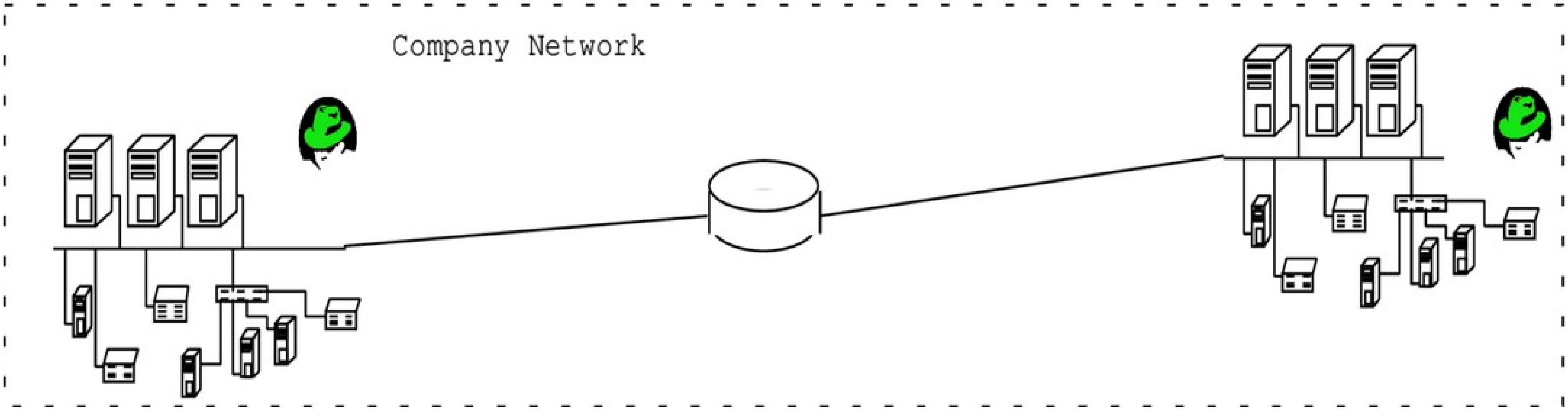
Virtual Private Networks (VPNs)



Acknowledgments: diagram by Utz Roedig at Lancaster University

Virtual Private Networks (VPNs)

Application Layer
4 Transport
3 Network
2 Data Link
1 Physical



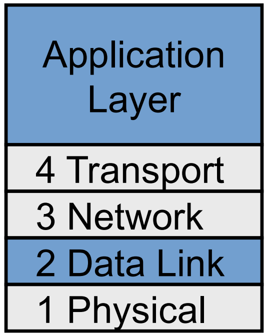
Acknowledgments: diagram by Utz Roedig at Lancaster University

Application Layer
4 Transport
3 Network
2 Data Link
1 Physical

Point-to-Point Tunneling Protocol (PPTP)

- Built-in on many clients, including Windows and MacOS/X
- Today used mostly for Internet ADSL dial-in
- Based on Point-to-Point Protocol (PPP) to transport network layer (layer 3) packets
- PPP also used for
  - remote address handling
  - user authentication via CHAP (challenge-response)
  - encryption via MPPE (RC4 based)
  - ➔ **Well-known to be insecure, don't use as a secure channel protocol!**
- Used channels
  - TCP control channel (port 1723) for tunnel set-up
    - no authentication, no encryption, no security
  - GRE data channel for transporting PPP packets
    - PPP packets transport content

Layer 2 Tunneling Protocol (L2TP)

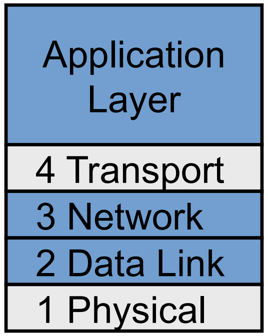


- Standardized in RFC 2661
- Combination of features from
  - Layer 2 Forwarding (L2F) designed by Cisco
  - Point-to-Point Tunneling Protocol (PPTP) designed by Microsoft
- L2TP comparable to PPTP, but:
  - can be used on arbitrary packet-switched networks (not only IP)
  - smaller header ⇒ less overhead
  - additional (optional) authentication of tunnel
  - supports multiple tunnels for load balancing
  - **typically used in combination with IPsec for security**
- Used channels
  - IP/UDP control channel
    - tunnel set-up, no encryption, but optional CHAP based authentication
  - IP/UDP data channel
    - PPP for content

Secure Socket Tunneling Protocol (SSTP)

- Proprietary Microsoft protocol, not available on other platforms by default (only through third-party clients)
- Uses standard TLS for secure channel handling, including default port 443
- Doesn’t directly support site-to-site tunneling, but focused on single clients
- Only supports user authentication, no device/network auth
- Always uses TCP for underlying packet transport
  - generally well supported through NAT (Network Address Translation) gateways
  - **but:** IP-over-TCP wrapping has performance (especially latency) issues when outer TCP connection requires retransmits

OpenVPN

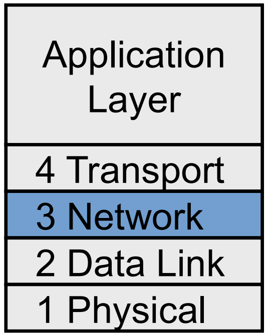


- Stand-alone VPN protocol with one reference implementation
  - available for most UNIX OS (including Linux, *BSD, MacOS), Windows, Android, etc.
- **Flexible network use**
  - can be used over TCP or UDP (both on port 1194 by default, but can use any port), through HTTP and SOCKS proxies, can *coexist with HTTPS service* on same port → advantages/disadvantages in TCP and UDP, can choose per scenario
  - due to standard TCP/UDP use, can easily go through NAT
  - typically used for „road warrior“ scenario (host-to-network), but can also be used for network-to-network VPN
  - can use either „tun“ (layer 3) or „tap“ (layer 2) virtual network devices → either virtual bridge or virtual router from a network point of view
- Security
  - keying inspired by TLS
  - authentication via static key, with X.509 certificates, and/or username/password
  - secure channel / packet format inspired by ESP (IPsec)
  - not standardized, but **currently assumed to be one of the more secure protocols** next to IPsec, TLS, and Wireguard (see e.g. OpenVPN use by Dutch government's national communications security agency, https://openvpn.fox-it.com/)

Application Layer
4 Transport
3 Network
2 Data Link
1 Physical

Wireguard

- Currently most modern protocol design
  - only fixed primitives (Curve25519, ChaCha20-Poly1305, BLAKE2)
    - simple to configure because cryptography negotiation non-existent
    - but might need new protocol versions in the future for agility
  - implemented as Linux kernel module, **fast** without hardware support
  - protocol properties have been **formally proven**
    (also see https://www.wireguard.com/papers/kobeissi-bhargavan-noise-explorer-2018.pdf)
- Routing only of IP packets, not data link layer
  - based on IP subnets or single target addresses configured at nodes
  - supports NAT keep-alive packets
  - supports transparent roaming of node IP addresses
- Authentication only with simple public keys (no user accounts)
  - a bit like SSH public keys (single line, ASCII encoded)
  - exchange of keys requires out-of-band channel
    - “left to the administrator”



IP Security (IPsec)

- Ggeneral IP Security mechanisms
- Provides
  - (data origin) authentication
  - confidentiality
  - connectionless integrity (with window based replay protection)
  - key management
- Applicable to use over LANs, across public and private WANs, and for the Internet
- Need identified in 1994 report, first specification in 1998
  - need authentication, encryption in IPv4 and IPv6
  - originally specified for IPv6, later adapted for IPv4
- Current RFCs: 4301-4303, 2407-2409, 4306 + many more
- Continuously updated and new features being developed

⇒ Currently one of the secure, but the most complex VPN standard!

IPsec evaluation

Advantages

- Interoperable between different vendors
- Is below transport layer, hence transparent to applications
- Can be transparent to end users
- **Can be fast** (close to wire speed) with hardware support
- **Can be highly secure and flexible** (if configured correctly)

Disadvantages

- Not as interoperable in practice
- Highly complex, historically grown protocol with too many options
- **Hard to configure**, can be used insecurely

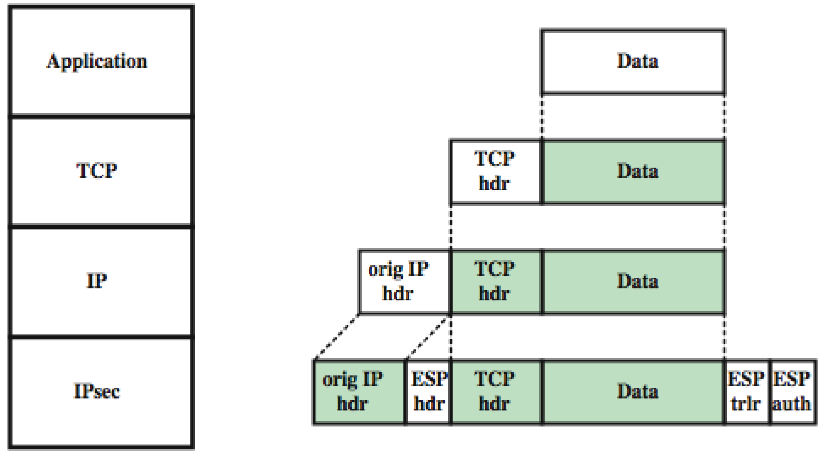
IPsec architecure

- Specification is quite complex, with groups:
  - 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*
  - cryptographic algorithms
  - and others...

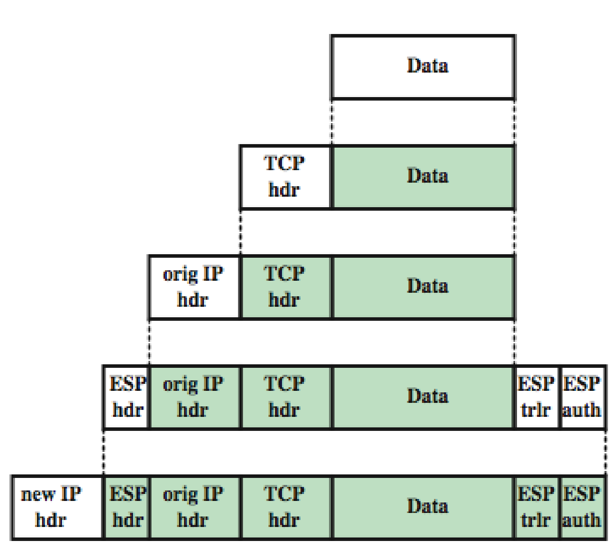
Transport and tunnel modes

- Transport Mode
  - host-to-host traffic / end-to-end security
  - to encrypt and optionally authenticate IP data
  - efficient in terms of overhead
  - attackers can do traffic analysis
  - can (with minor differences) be regarded as a sub-set of tunnel mode
    - criticized for causing unnecessary complexity in standard
- Tunnel Mode
  - network-to-network, host-to-network, or host-to-host (VPNs)
  - encrypts entire IP packet
  - add new header for next hop ⇒ next header field is major difference between transport and tunnel modes
  - no routers on way can examine inner IP header

Transport and Tunnel Mode Protocols



(a) Transport mode



(b) Tunnel mode

IPsec protocols

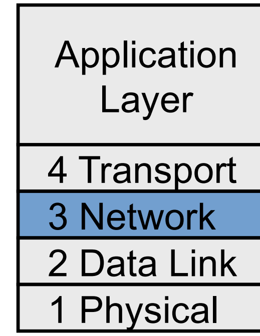
Remember modes, protocols, and relationship!

- ESP: Encapsulating Security Payload
  - IP protocol number 50
    - “protocol” = same level as IP, ARP etc. this is not a port number!
  - (optional) **authentication and encryption of payload**
- AH: Authentication Header
  - IP protocol number 51
  - **only authentication, but payload + IP header**
  - all IP header fields with the exception of TOS, flags, fragment offset, TTL, and header checksum included in authentication
- Common to both channel protocols:
  - **IKE** (Internet Key Exchange) for key management, builds upon
  - **ISAKMP**
- Typical combinations
  - tunnel mode + ESP
  - transport mode + ESP with L2TP in IPsec tunnel
  - transport mode + AH

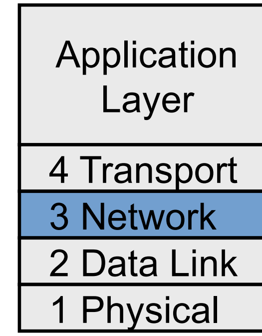
Security Associations (SAs)

- A one-way relationship between sender and receiver that affords security for traffic flow
- Defined by 3 parameters:
  - Security Parameters Index (SPI)
  - IP Destination Address
  - Security Protocol Identifier
- Has a number of other parameters
  - sequence number, AH and EH info, lifetime etc
- Have a database of Security Associations

Encapsulating Security Payload (ESP)

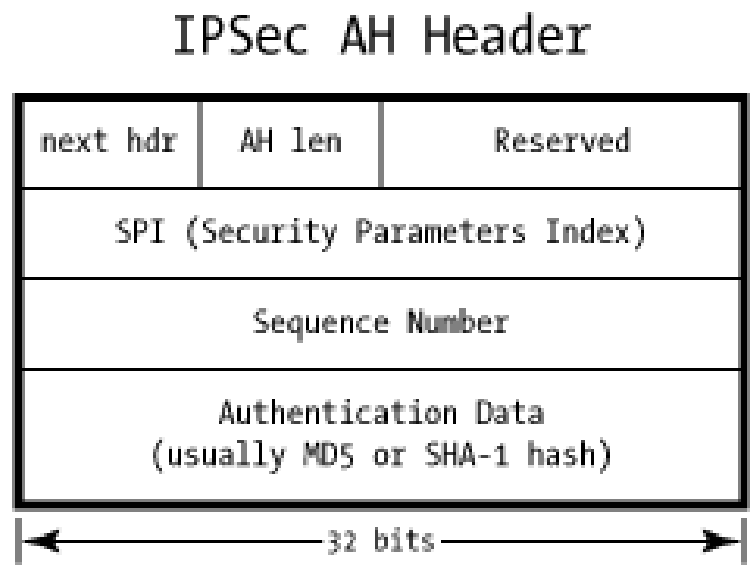


- Provides message content confidentiality, data origin authentication, connectionless integrity, an anti-replay service, limited traffic flow confidentiality
  - sender initializes sequence number to 0 when a new SA is established, increment for each packet, must not exceed limit of $2^{32} - 1$
  - receiver then accepts packets with seq no within window of ($N - W + 1$)
- Services depend on options selected when establishing Security Association (SA), network location
- Can use a variety of encryption and authentication algorithms
- Can be used with transport or tunnel mode (distinction with next header field)
- Can be used with NAT-traversal

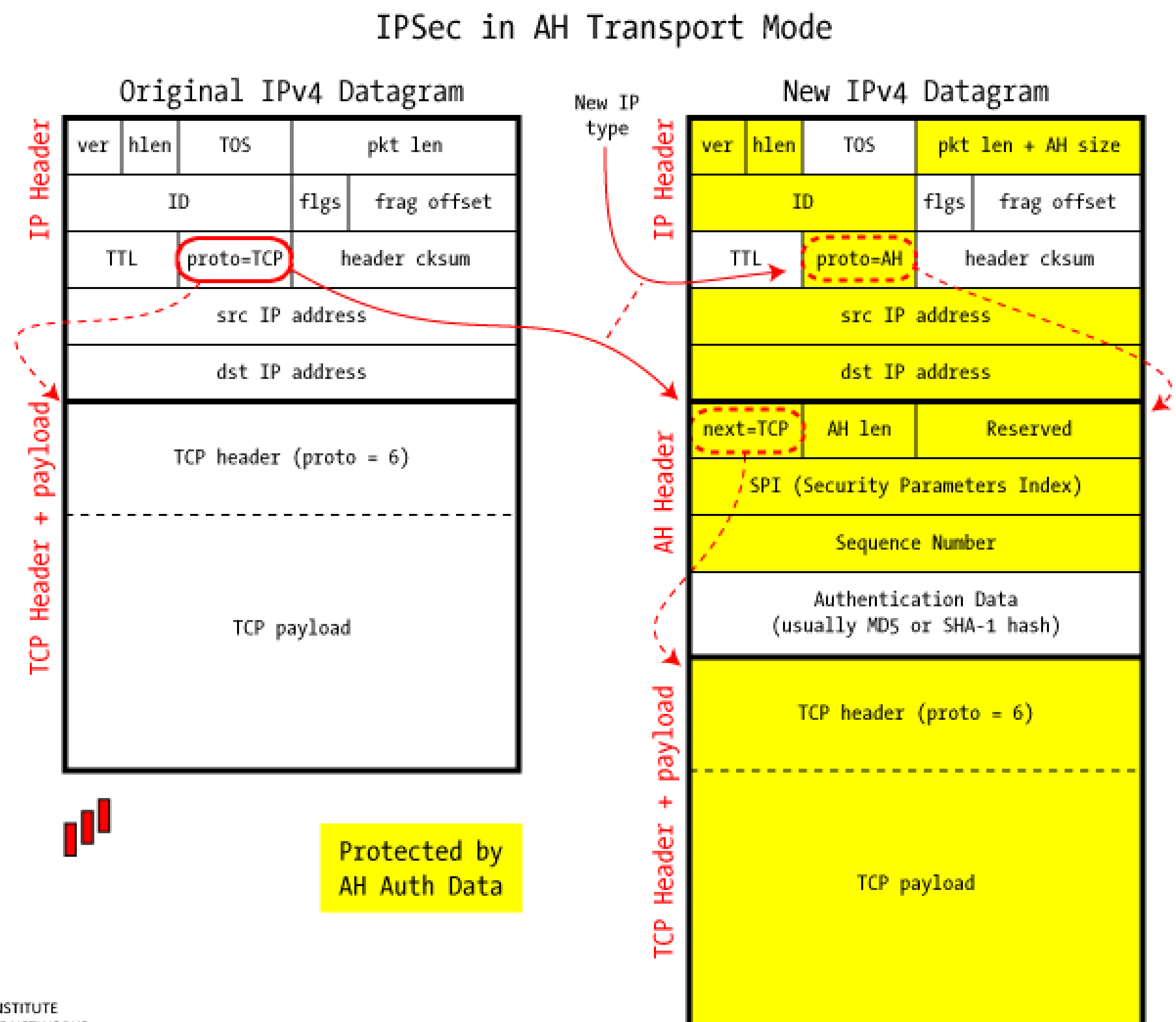


Authentication Header (AH)

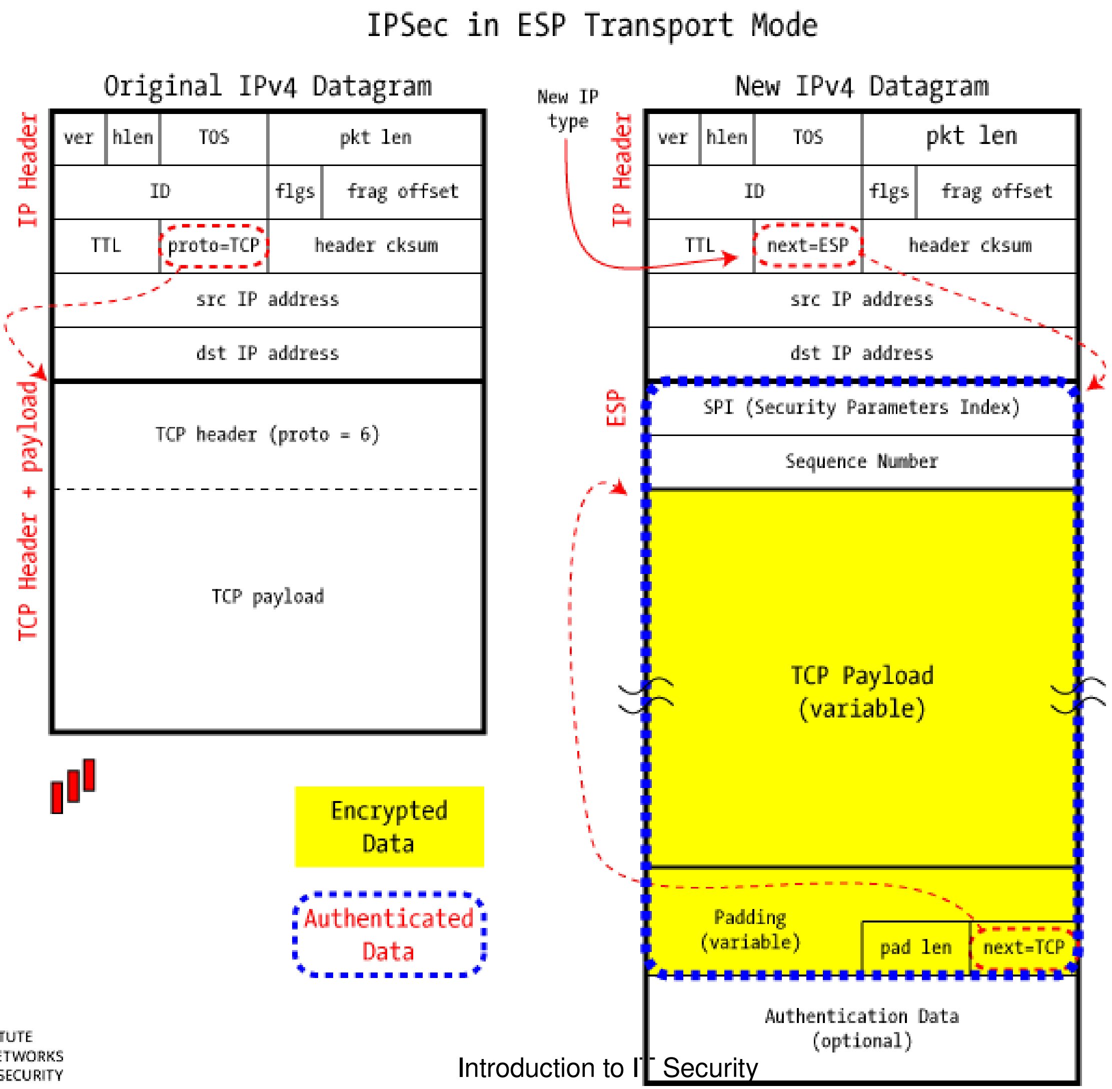
- Length of authentication data variable to support use of different algorithms
- Why AH when we already have ESP?
  - to authenticate outer header in tunnel mode or the only IP header in transport mode (ESP does not protect outer header!)
  - slightly less overhead
  - for IPv6 only ESP is mandatory, AH declared optional



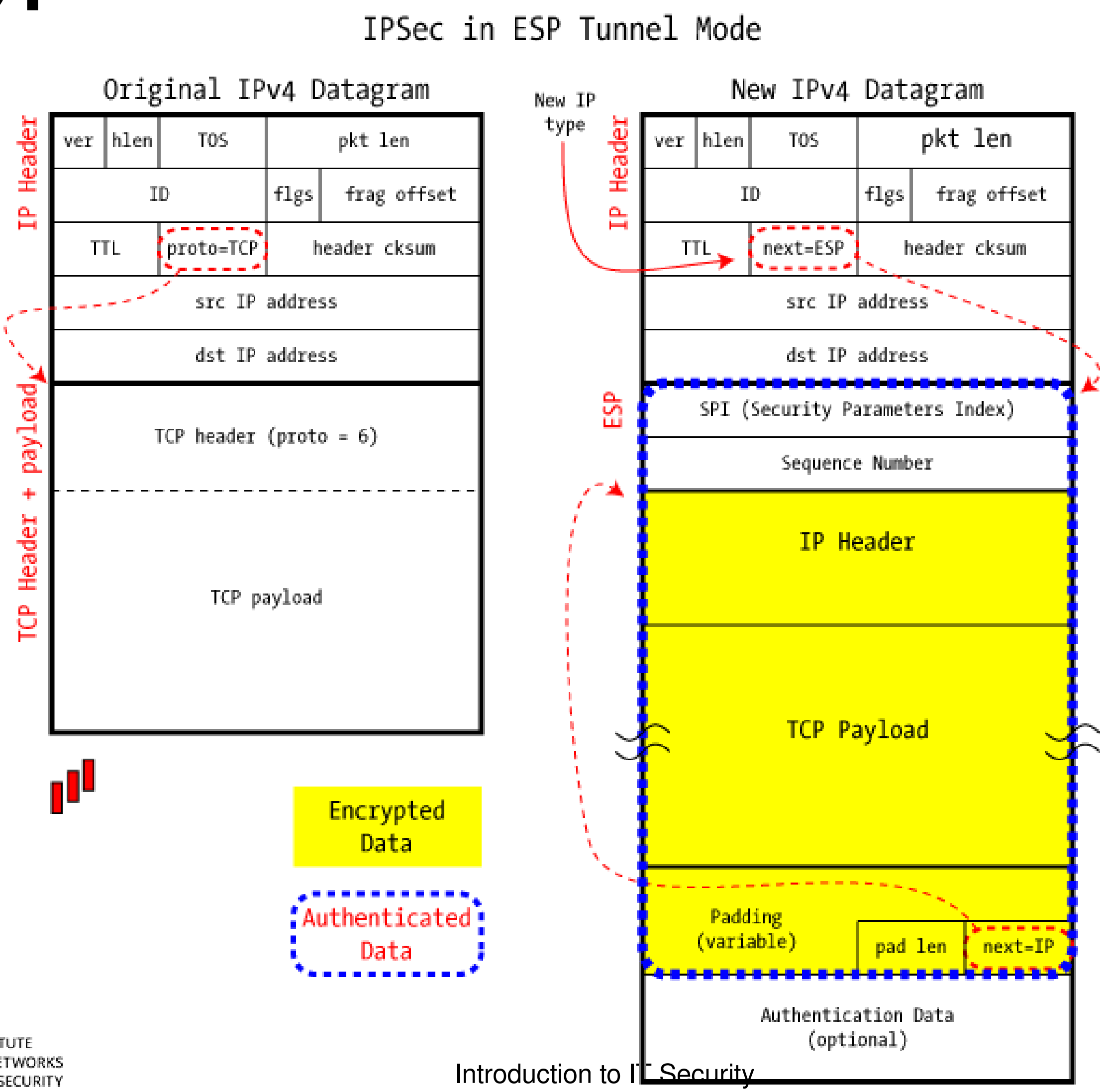
IPsec: typical combinations



IPsec: typical combinations



IPsec: typical combinations



IPsec key management

- Handles key generation and distribution
- Typically need 2 pairs of keys
  - 2 per direction for AH and ESP
- Manual key management
  - sysadmin manually configures every system
- Automated key management
  - automated system for on demand creation of keys in large systems
  - has Oakley and ISAKMP elements

Oakley

- a key exchange protocol
- Based on Diffie-Hellman key exchange for **Perfect Forward Secrecy (PFS)**
- UDP port 500 or 4500 for NAT-traversal
- Adds features to address weaknesses
  - no info on parties, man-in-middle attack, cost
  - so adds cookies, groups (global params), nonces, DH key exchange with authentication
- Can use arithmetic in prime fields or elliptic curve fields
- Authentication
  - authentication based on hosts, not users
  - authentication always mutual
  - standard options:
    - Pre-Shared Key (PSK), comparable to password
    - RSA public/private key, typically with X.509 PKI
    - optional extensions for user authentication (XAUTH), or use with L2TP

ISAKMP

- Internet Security Association and Key Management Protocol (ISAKMP)
- Provides framework for key management
- Defines procedures and packet formats to establish, negotiate, modify, and delete SAs
- Independent of key exchange protocol, encryption algorithm, and authentication method
- IKEv2 no longer uses Oakley and ISAKMP terms, but basic functionality is same

IPsec keying protocol phases

- IKEv1 messages and phases:
  - IKE phase 1: Main Mode (MM), negotiates ISAKMP SA (aka IKE SA), based on DH and authentication (e.g. PSK or X.509/RSA)
  - IKE phase 2: Quick Mode (QM): negotiates IPsec SA(s) (mode, protocol(s), algorithms, keys), secured by ISAKMP SA
- Periodic re-keying of both ISAKMP SA and IPsec SA
  - IPsec SA more often than ISAKMP SA
    - Why? Name 2 reasons!
- IKEv2 similar to IKEv1, slightly optimized, better support for QoS, support for error messages, support for MobIKE

IPsec glossary

- AH — Authentication Header
- AM — Aggressive Mode (faster connection establishment, but weak privacy guarantees, therefore not recommended)
- ESP — Encapsulating Security Payload
- MM — Main Mode (more security than Aggressive Mode, but 6 instead of 3 packets)
- PFS — Perfect Forward Secrecy
- QM — Quick Mode
- DH — Diffie-Hellman
- IKE — Internet Key Exchange
- ISAKMP — Internet Security Association and Key Management Protocol
- SPI — Security Parameters Index
- SA — Security Association
- SAD(B) — Security Associations DataBase
- SPD(B) — Security Policy DataBase

IEEE 802.11 security

- Wireless traffic can be monitored by any radio in range, not physically connected
- Original 802.11 spec had security features
  - **Wired Equivalent Privacy (WEP)** algorithm
  - but found this contained major weaknesses → **DON'T USE!**
- 802.11i task group developed capabilities to address WLAN security issues
  - Wi-Fi Alliance **Wi-Fi Protected Access (WPA)**
  - final 802.11i **Robust Security Network (RSN)**
    - Wi-Fi Alliance also uses term **WPA2** to refer to the use of CCMP (AES)
    - finalized **WPA3** standard in 2018 with **improvements** to maximum security level (192 instead of 128 bits), initial key exchange in personal mode, forward secrecy, and protecting management frames (e.g. deauth)
      → potentially biggest improvement is **encryption of open network traffic**

Extensible Authentication Protocol (EAP)

- Standardized as RFC 3748
  - not specific to WLAN, but can be used within IEEE 802.11i, encapsulated with IEEE 802.1x
  - framework for network access and authentication protocols
  - can operate over different network and link level protocols
- Supports multiple authentication methods:
  - EAP-TLS (RFC 5216): mutual authentication with certificates
  - EAP-TTLS (tunneled TLS, RFC 5281): server authenticates via certificate, client with other EAP method oder legacy PAP/CHAP (username/password) – may have security issues
  - EAP-IKEv2 (RFC 5106): uses IKEv2 authentication methods
  - EAP-GPSK (RFC 5433): using pre-shared key (PSK), uses only symmetric cryptography
  - **PEAP** (protected EAP): like EAP-TTLS, server authenticates via certificate, client with other EAP method (username/password), often used for WLAN with WPA2/RSN in configurations PEAPv0/EAP-MSCHAPv2 (common) or PEAPv1/EAP-GTC (rare)
  - EAP-SIM (RFC 4186): uses existing SIM card authentication protocols
  - **EAP-AKA** (RFC 4187): uses UMTS authentication via USIM
  - EAP-EKA (RFC 6124): new mode based on Diffie-Hellman with only short passwords and without certificates,

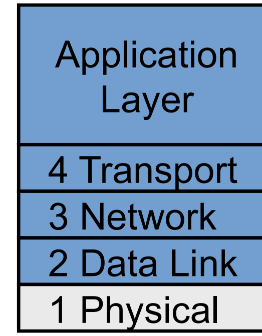
802.11i
protected data transfer phase

- Have two schemes for protecting data
- Temporal Key Integrity Protocol (TKIP)
  - s/w changes only to older WEP
  - adds 64-bit Michael message integrity code (MIC)
  - encrypts MPDU plus MIC value using RC4
    ⇒ called WPA (either WPA-PSK or WPA Enterprise)
  - **don't use anymore!**
- Counter Mode-CBC MAC Protocol (CCMP)
  - CCM mode uses the cipher block chaining message authentication code (CBC-MAC) for integrity
  - uses the CTR block cipher mode of operation
    ⇒ called WPA2 (either WPA2-PSK or WPA2 Enterprise or RSN)
- WPA3: better authentication (only one password try; brute-force more difficult), PFS, secure integration of display-less devices via a third one

WPA3 - EasyConnect

- **Problem**: device without display/keyboard
  - How to integrate it securely? DH key exchange + verify identity
  - But how without keyboard/display?
- **Solution**:
  - sticker (scan QR-code) on both device (“Enrollee”) and router
  - scan both stickers with an App on a mobile phone (“Configurator”)
    - or enter a human-readable string, i.e. a “secret key”
  - phone then sends configuration parameters to device
  - device then securely connects to router
- Security:
  - Is this really the original sticker with the real QR code?
  - App knows the device, but how does the device know the App?

(Physical, local, spontaneous) Device-to-device authentication

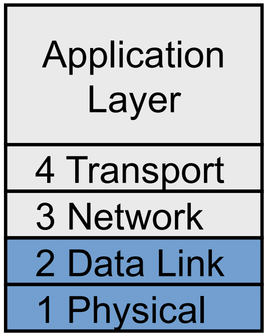


- Currently a lot of communication happens directly between two (or multiple) devices in close proximity
  - these often communicate wirelessly
  - transport security of communication is desired, therefore need to establish secure channel
  - first contact often spontaneous / serendipitous → no admin
- Main problem is **authentication without relying on third parties**
- Want to provide **Perfect Forward Secrecy (PFS)** to safeguard against future leaking of long-term secrets
- Want to force attackers into **active online attacks** instead of passive brute-force attacks

Authentication of wireless channels

Typical approach for secure channel setup:

- Key agreement: typically select peer device + (EC-) Diffie-Hellman
- Peer authentication: various options
  - commitment schemes
  - interlock-based protocols
- Verification based on some out-of-band channel
  - verification of key hashes: display+user+yes/no
  - transmission over secret and/or authentic channel: display+user+keypad, infrared, ultrasound, laser, display+camera, audio, NFC, ...
  - shared secret: common data, possibly “fuzzy”



Security properties of out-of-band channels

Out-of-band channels can be

- confidential
- stall-free
- authentic (most useful property to have)
- or provide partial integrity

or any combination

Recent protocol proposals: standards based on MANA-IV

- [S. Laur and K. Nyberg: "*Efficient Mutual Data Authentication Using Manually Authenticated Strings*", CANS 2006]
- Bluetooth pairing in current standard and WLAN WEP are completely broken

[Y. Shaked and A. Wool: "Cracking the Bluetooth PIN", Mobisys 2005]

[F.-L. Wong, F. Stajano, and J. Clulow: "Repairing the Bluetooth pairing protocol", Security Protocols 2005]

[E. Tews, R.-P. Weinmann, and A. Pyshkin: "Breaking 104 bit WEP in less than 60 seconds", Cryptology ePrint Archive 2007/120]

- Bluetooth Simple Pairing [Bluetooth SIG: Simple Pairing Whitepaper, 2006]
  - "just works" - insecure against MITM
  - "numeric comparison" of six digit number, yes/no on both devices
  - "out of band" e.g. with NFC
  - "passkey entry" with transferring a six digit number (human as out-of-band channel)
- Wi-Fi Protected Setup (WPS)
  - "push button configuration" - insecure against MITM
  - "PIN" with four to eight digit number
  - "out-of-band" e.g. with NFC

Remark: What to do after device authentication?

- Devices also need internal state and key management
- e.g. “Resurrecting Duckling”
  [F. Stajano and R. Anderson: “The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks”, 7th Workshop on Security Protocols, 1999]
  - Device trusts the first thing it sees on “birth” and accepts it as owner (password, public key, etc.)
    → Reset device for a new “birth” to connect it to attacker (or extract key...)
- Key storage
  - securing keys against physical access
  - securing keys in memory
  - deleting keys
- Trust
  - building trust (user assigned, reputation approaches)
  - revoking trust
  - trust delegation
- **Without a public key infrastructure**