SSL AND IPSEC

GOALS

- Security in practice
  - In transport layer
  - In network layer

SECURING TCP CONNECTIONS: SSL

Transport Layer Security (TLS)

SSL: SECURE SOCKETS LAYER

- Widely deployed security protocol
  - Supported by almost all browsers, web servers
  - HTTPS
  - Billions $/year over TLS

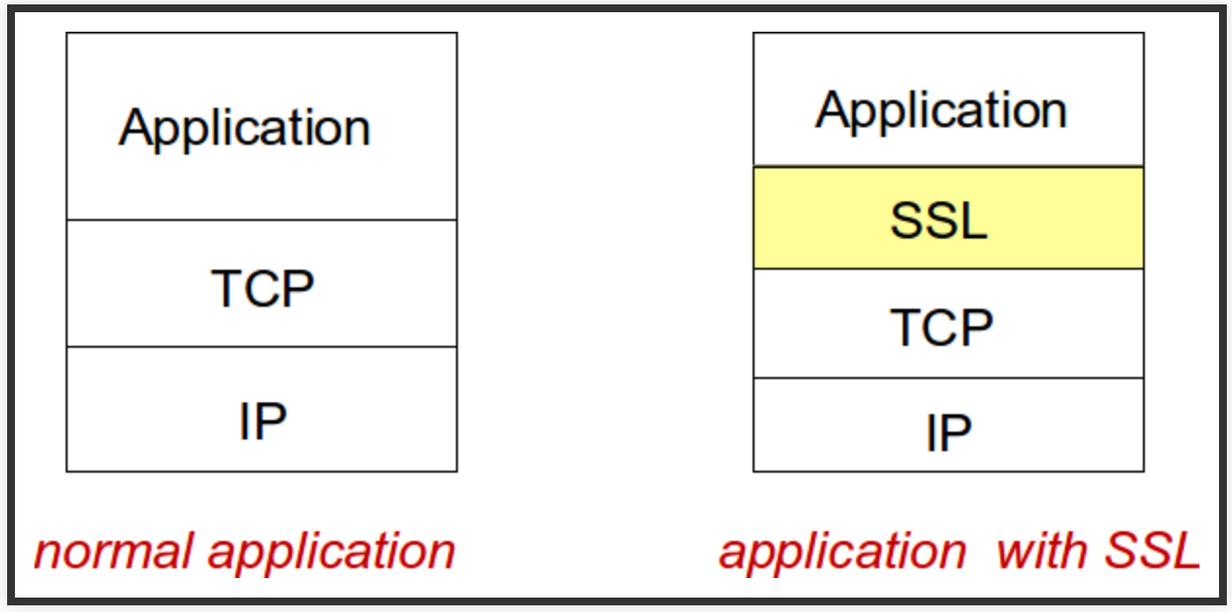
SSL: SECURE SOCKETS LAYER

- Mechanisms: [Woo 1994], implementation: Netscape
- Variation - TLS: transport layer security, RFC 2246
- Provides
  - **Confidentiality**
  - **Integrity**
  - **Authentication**

SSL: SECURE SOCKETS LAYER

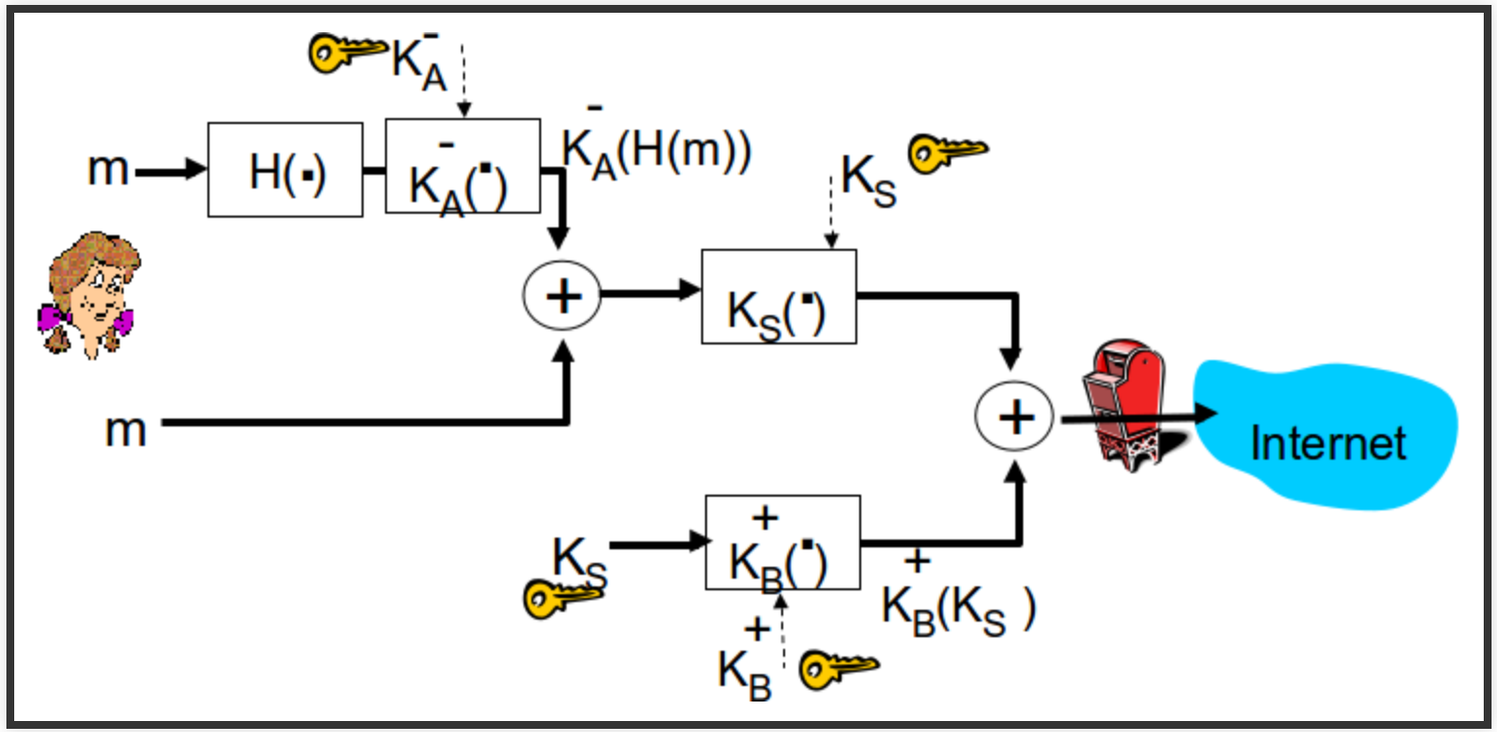
- Original goals:
  - Web e-commerce transactions
  - Encryption (especially credit-card numbers)
  - Web-server authentication
  - Optional client authentication
  - Minimum hassle in doing business with new merchant
- Available to all TCP applications → secure socket interface

SSL AND TCP/IP



- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

COULD DO SOMETHING LIKE PGP



COULD DO SOMETHING LIKE PGP - BUT

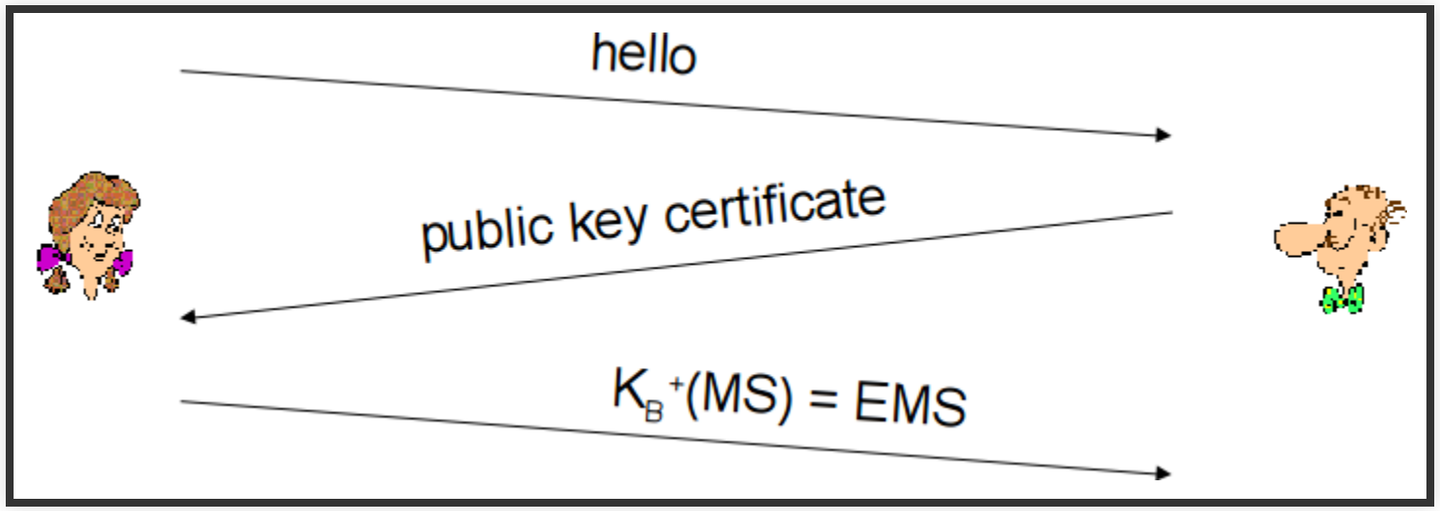
- Want to send byte streams and interactive data
- Want set of secret keys for entire connection
- Want certificate exchange as part of protocol: **Handshake phase**

TOY SSL (TSSL): A SIMPLE SECURE CHANNEL

TSSL

- **Handshake:** Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- **Key derivation:** Alice and Bob use shared secret to derive set of keys
- **Data transfer:** data to be transferred is broken up into series of records
- **Connection closure:** special messages to securely close connection

TSSL: HANDSHAKE



- **MS:** master secret
- **EMS:** encrypted master secret

TSSL: KEY DERIVATION

- Considered bad to use same key for more than one cryptographic operation
  - Use different keys for message authentication code (MAC) and encryption

TSSL: KEY DERIVATION

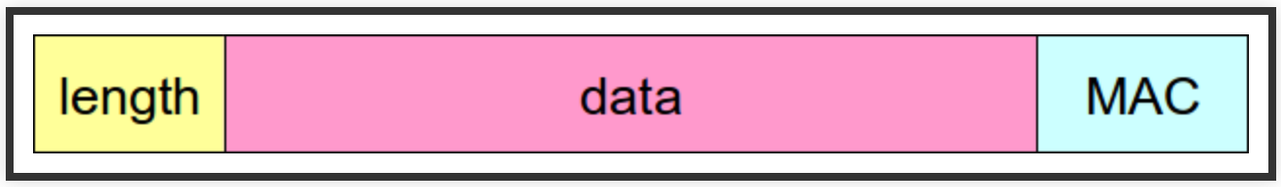
- **Four** keys:
  - $\mathbf{K_C}$ = encryption key for data sent from client to server
  - $\mathbf{M_C}$ = MAC key for data sent from client to server
  - $\mathbf{K_S}$ = encryption key for data sent from server to client
  - $\mathbf{M_S}$ = MAC key for data sent from server to client
- Keys derived from key derivation function (KDF)
  - Takes master secret and (possibly) some additional random data and creates the keys

TSSL: STREAM OR RECORDS

- Why not encrypt data in constant stream as we write it to TCP?
  - Where would we put the MAC? If at end, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- Instead, break stream in series of records
  - Each record carries a MAC
  - Receiver can act on each record as it arrives

TSSL: DATA AND MAC

- **Issue:** in record, receiver needs to distinguish MAC from data
  - Want to use variable-length records



TSSL: HANDLE REPLAY ATTACKS

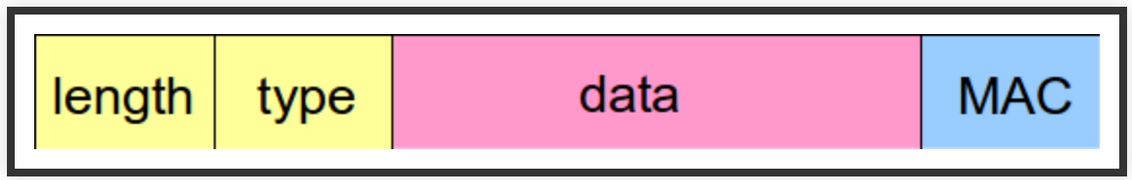
- **Problem:** attacker can capture and replay record or re-order records
- **Solution:** put sequence number into MAC:
  - MAC = MAC(M_X, sequence || data)
  - Note: no sequence number field

TSSL: FULL REPLAY ATTACK

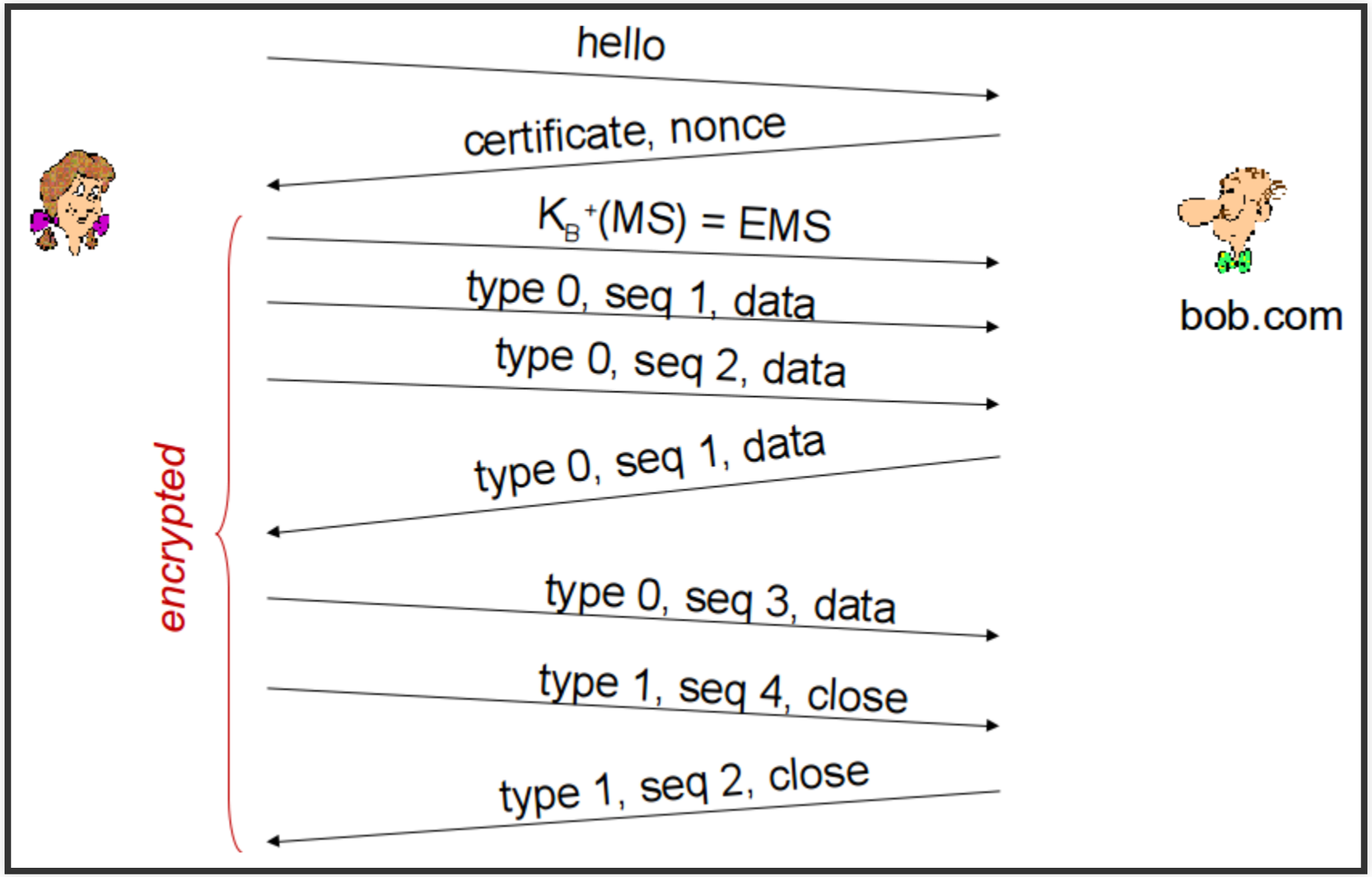
- **Problem:** Attacker could replay all records
- **Solution:** Use nonce

TSSL: HANDLE TRUNCATION ATTACK

- **Problem:** Truncation attack:
  - Attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is.
- **Solution:** Add **record types**, with one type for closure
  - Type 0 for data; type 1 for closure
- MAC = MAC(M_X, sequence || type || data)



TOY SSL (TSSL): SUMMARY



TOY SSL ISN'T COMPLETE

- How long are fields?
- Which encryption protocols?
- Want negotiation?
  - Allow client and server to support different encryption algorithms
  - Allow client and server to choose together specific algorithm before data transfer

SSL CIPHER SUITE

SSL CIPHER SUITE

- Cipher suite
  - Public-key algorithm
  - Symmetric encryption algorithm
  - MAC algorithm
- SSL supports several cipher suites
- Negotiation: client, server agree on cipher suite
  - Client offers choice → server picks one

SSL CIPHER SUITE

Common SSL **symmetric ciphers**

- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

SSL **Public key encryption**

- RSA

REAL SSL: HANDSHAKE (1)

Purpose

- Server authentication
- Negotiation: agree on crypto algorithms
- Establish keys
- Client authentication (optional)

REAL SSL: HANDSHAKE (2)

- Client sends list of algorithms it supports, along with client nonce
- Server chooses algorithms from list; sends back: choice + certificate + server nonce
- Client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
- Client and server independently compute encryption and MAC keys from pre_master_secret and nonces
- Client sends a MAC of all the handshake messages
- Server sends a MAC of all the handshake messages

REAL SSL: HANDSHAKING (3)

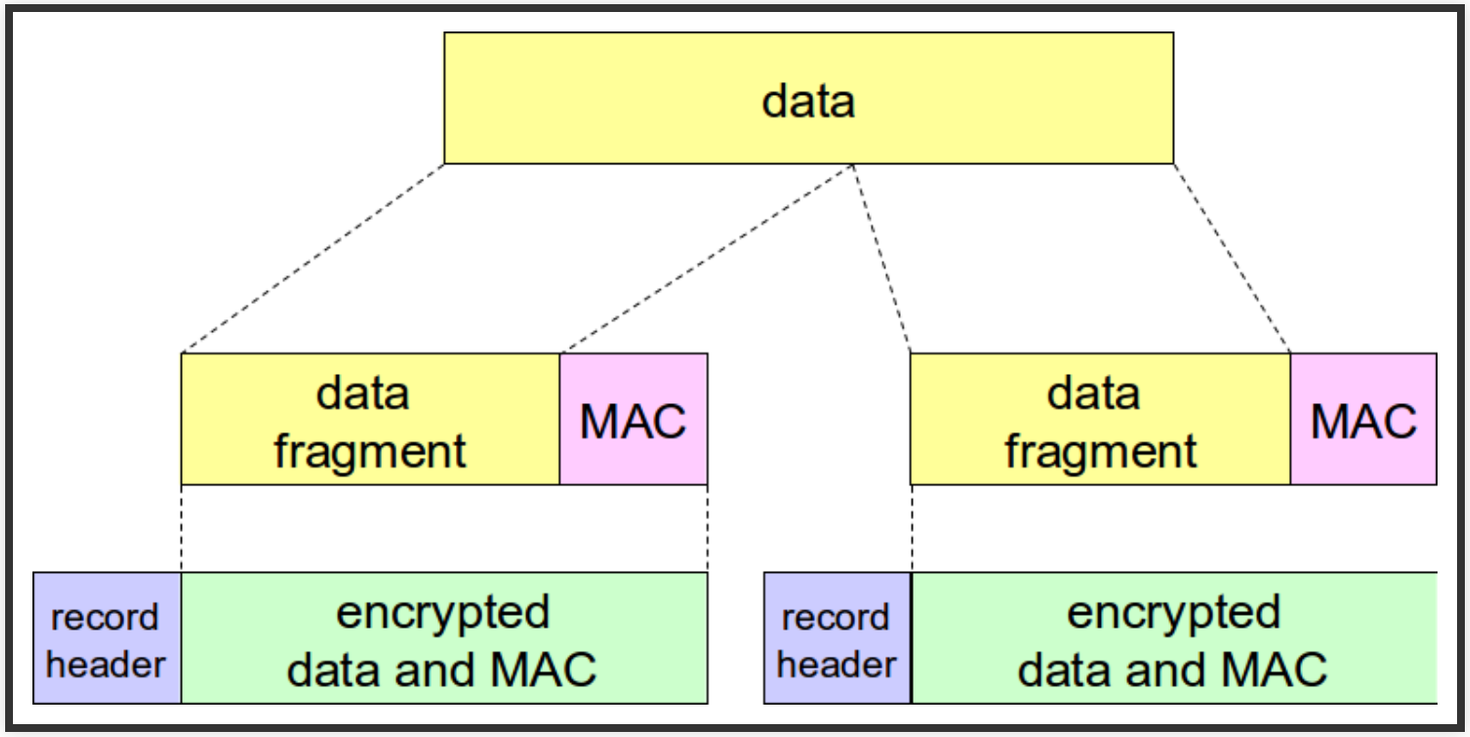
Last 2 steps protect handshake from tampering

- Client typically offers range of algorithms, some strong, some weak
- Man-in-the middle could delete stronger algorithms from list
- Last 2 steps prevent this
  - Last two messages are encrypted

REAL SSL: HANDSHAKING (4)

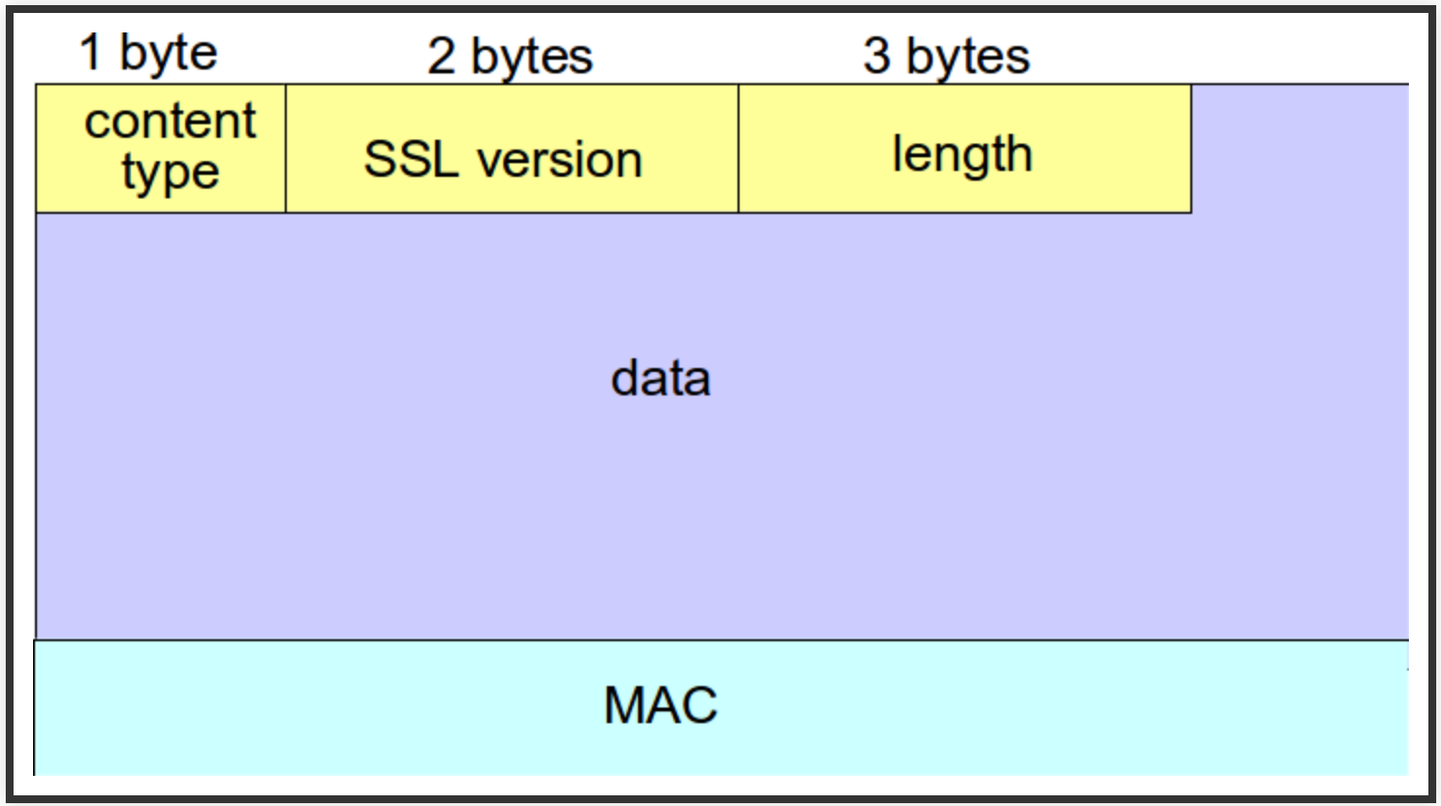
- Why two random nonces?
- Suppose Trudy sniffs all messages between Alice & Bob
- Next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - Solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

SSL RECORD PROTOCOL



- **Record header:** content type; version; length
- **MAC:** includes sequence number, MAC key M_X
- **Fragment:** each SSL fragment 2^{14} **bytes** (~16* Kbytes)

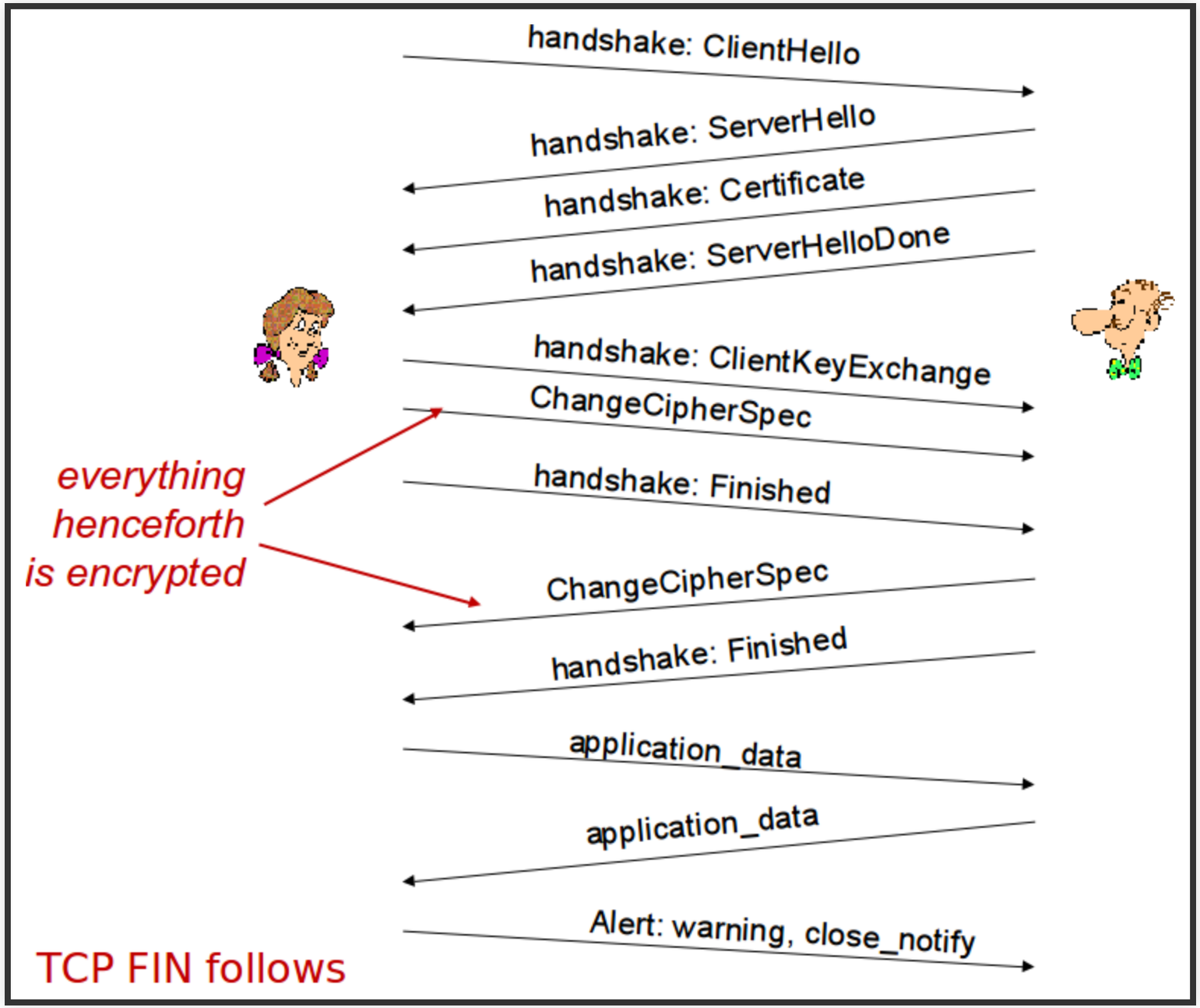
SSL RECORD FORMAT



Data and MAC encrypted (symmetric algorithm)

REAL SSL CONNECTION

REAL SSL CONNECTION



KEY DERIVATION

- Client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - Produces master secret
- Master secret and new nonces input into another random-number generator: “key block”

KEY DERIVATION

Key block sliced and diced:

- Client MAC key
- Server MAC key
- Client encryption key
- Server encryption key
- Client initialization vector (IV)
- Server initialization vector (IV)

TLS VS. SSL

- SSL v3.0 laid the foundation for TLS v1.0
- Both use similar ciphers and message digests (though there are newer and better ciphers that are only available with newer versions of TLS such as TLS v1.1 and or v1.2).
- Differences in how the secure communications are established are also evident and make TLS v1.0 (and higher) clearly stronger than SSL v3.0.
- Which protocol is used – TLS v1.x or SSL v3.0 – is determined by a negotiation between the client and server, based on what software is installed on each and how each is configured.

TLS VS. SSL

SSL v3.0

- Was exploited by the POODLE attack and is now obsolete

TLS v1.2

- The most widely-used TLS protocol
- Enables better use of more secure ciphers
- Features enhanced negotiation of the encrypted connections

Recommended to disable TLS 1.0 and 1.1

TLS VERSION 1.3

- https://tools.ietf.org/html/rfc8446.html
- All insecure algorithms removed
- zero round-trip time (0-RTT) mode was added
  - Lower security, but saves a RTT
- RSA and Diffie-Hellman cipher suites now provide forward secrecy
- All handshake messages after the ServerHello are now encrypted.
- Elliptic curve algorithms are now in the base spec

TLS VERSION TEST

Which version does SDU support?

Try https://www.ssllabs.com/ssltest

See Configuration → Protocols

ILLUSTRATED TLS CONNECTION

Very well documentation

https://tls.ulfheim.net/

SSL SECURITY BREACHES

- Freak
- Beast
- Poodle
- Heartbleed
- Zombie poodle, Goldendoodle, 0-Length OpenSSL, Sleeping poodle

FREAK

The **F**actoring **R**SA **E**xport **K**eys attack

Attack can harm millions of unaware users worldwide.

Forcing vulnerable clients to downgrade to weaker crypto

The FREAK attack is possible because of the fact that the United States has published a policy which would deny the use of stronger SSL/TLS cryptographic methods.

FREAK

The US limiting exportable software to use only public key pairs with RSA moduli of 512 bits or less (so-called RSA_EXPORT keys), with the intention of allowing them to be broken easily by the National Security Agency (NSA), but not by other organizations with lesser computing resources.

However, by the early 2010s, increases in computing power meant that they could be broken by anyone with access to relatively modest computing resources using the well-known Number Field Sieve algorithm, using as little as $100 of cloud computing services.

AN ATTACK

1. Find a vulnerable server that offers export cipher suites
    - it should reuse a key for a longish time
2. break key
3. find vulnerable client
4. attack via MITM (easy to do on a local network or wifi; not so easy otherwise).

> Good lesson here: Don't enable technologies that you don't want to see used, even if you don't really think they will be used.

CVE'S

- CVE-2015-0204
- CVE-2015-1637

BEAST

- May 2011
- Juliano Rizzo and Thai Duong released a paper: **Here Come The XOR Ninjas**
- Concrete proof of concept against implementations of SSL 3.0 and TLS 1.0 using cipher block chaining (CBC) encryption scheme in a browser environment.
- The BEAST (Browser Exploit Against SSL/TLS)
- Before the paper, attacks against CBC were believed to be theoretical.

BEAST

- In a SSL connection if the data is encrypted using CBC with chained IVs, it allows a man-in-the-middle to obtain plain text HTTP headers using blockwise-adaptive chosen-plaintext attack
  - In short it will allow decrypting HTTPS requests and steal information such as session cookies to be used to impersonate the user.
- The attack has been demonstrated by the researchers on the Ekoparty security conference using Java, a network sniffer and a popular browser

BEAST

Not easy to pull off ⇒ Low CVSS score of 4.3.

https://nvd.nist.gov/vuln/detail/CVE-2011-3389

POODLE

The **P**adding **O**racle **O**n **D**owngraded **L**egacy **E**ncryption attack

- Man in the middle attack
- Exploits the fallback to SSL 3.0.
- New variant announced on December 8, 2014.
  - exploits implementation flaws of CBC encryption mode in the TLS 1.0 – 1.2 protocols.
  - TLS specifications require servers to check the padding, some implementations fail to validate it properly

POODLE

To work with legacy servers, many TLS clients implement a downgrade dance: in a first handshake attempt, offer the highest protocol version supported by the client; if this handshake fails, retry (possibly repeatedly) with earlier protocol versions.

Unlike proper protocol version negotiation (if the client offers TLS 1.2, the server may respond with, say, TLS 1.0), this downgrade can also be triggered by network glitches, or by active attackers.

POODLE

⚠ Can end with SSL 3.0 = Insecure

CVE'S

- CVE-2014-3566
- CVE-2014-8730

Extended description:

- https://www.openssl.org/~bodo/ssl-poodle.pdf.

VULNERABLE?

If you want to know if you are vulnerable to the Poodle attack, then you can take a look at the following Poodle scanner:

- https://www.poodlescan.com/

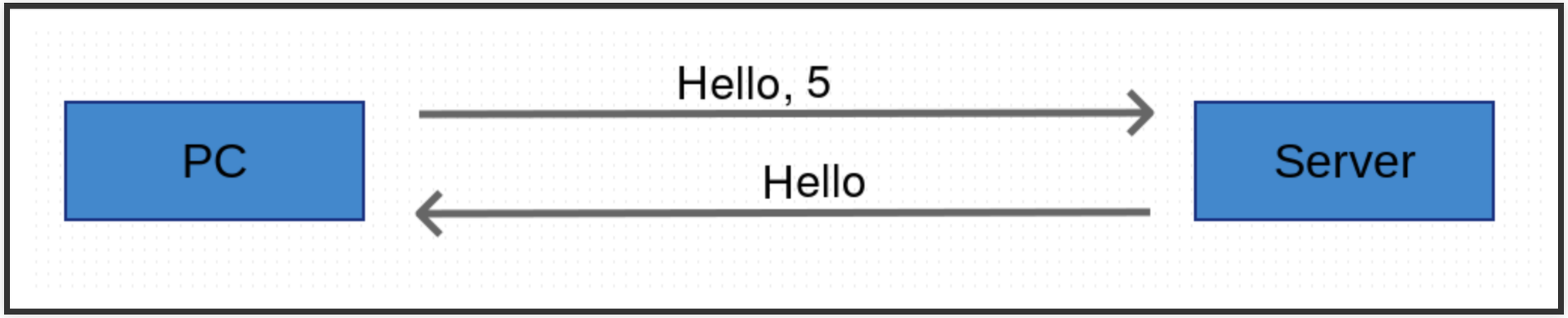
HEARTBLEED

- Buffer underflow attack
- Published in April 2014
- OpenSSL cryptography library
  - Globally used for the Transport Layer Security protocol.
- Gain access to the memory of the service which holds the secret key of the SSL/TLS communication.

> ⚠ Allow hackers to perform man in the middle attacks.

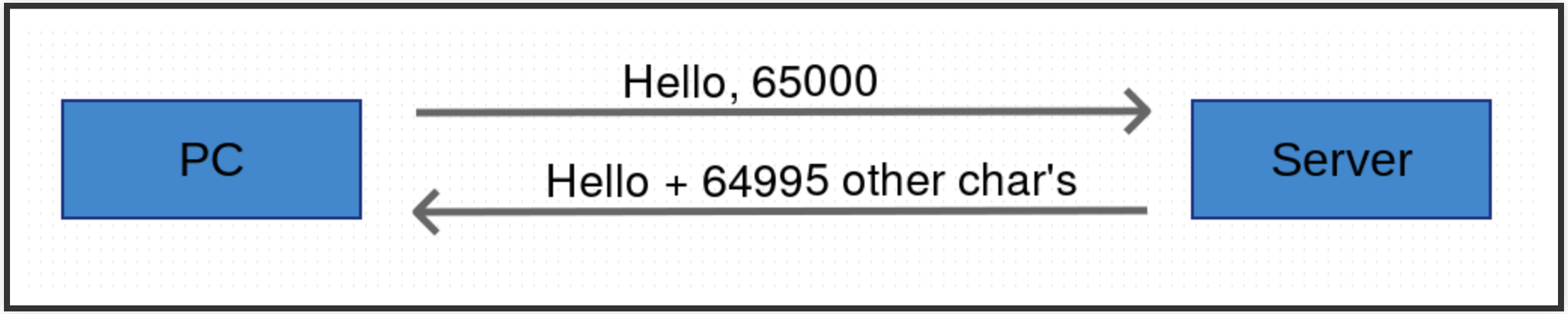
HOW

Heartbeet command



HOW

Heartbeet command



AFFECTED SERVICES

- Internet surfing
- E-mail communication
- Instant messaging communication
- Virtual Private Networks

CVE

- CVE-2014-0160

OTHER ATTACKS

- Zombie POODLE (Invalid padding with valid MAC)
- GOLDENDOODLE (Valid padding with an invalid MAC)
- 0-Length OpenSSL (Invalid Mac/Valid Pad, 0-length record)
- Sleeping POODLE (invalid padding with valid MAC)

OTHER ATTACKS

- https://www.tripwire.com/state-of-security/vert/zombie-poodle/
- https://www.tripwire.com/state-of-security/vert/goldendoodle-attack/
- https://nvd.nist.gov/vuln/detail/CVE-2019-1559

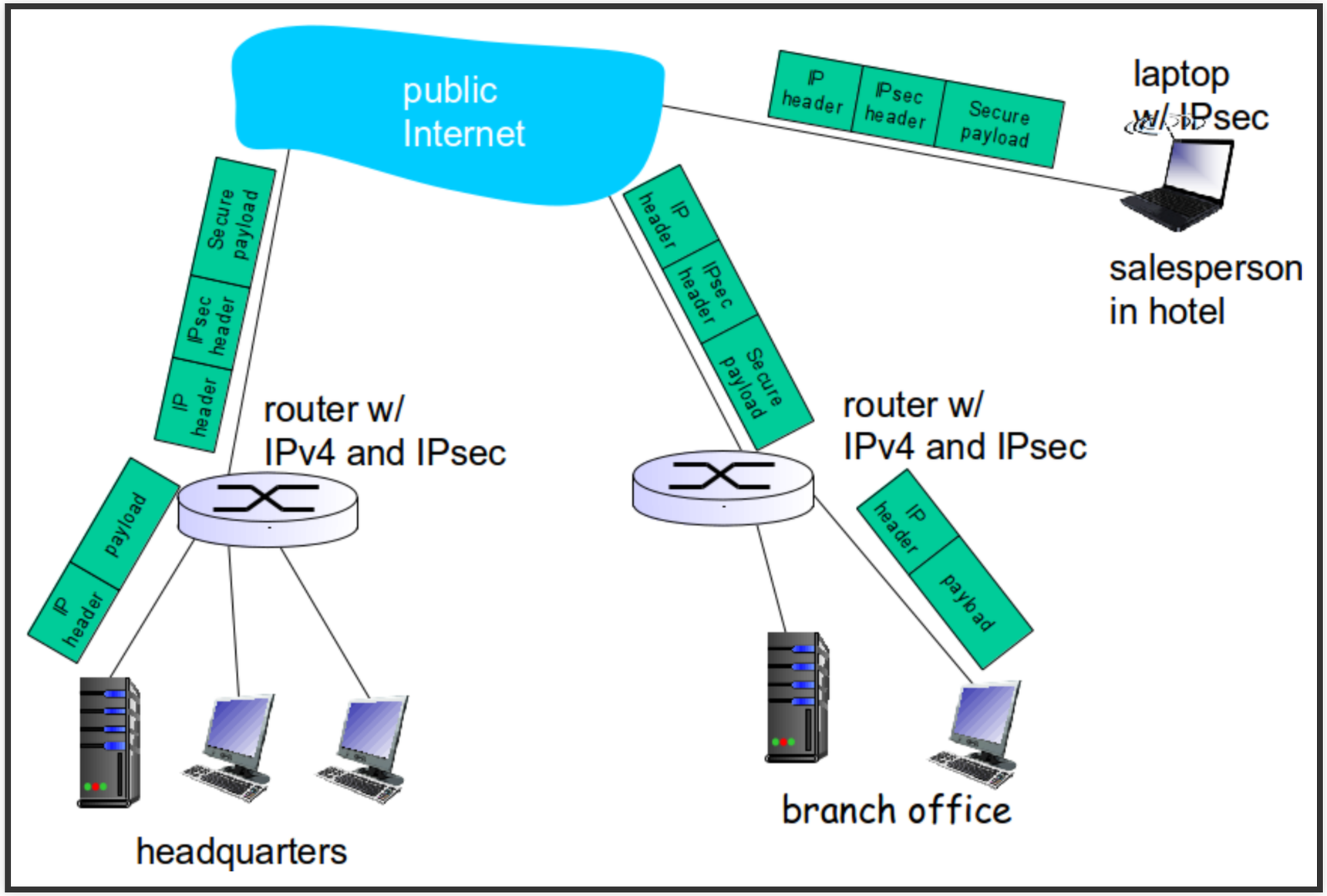
NETWORK LAYER SECURITY: IPSEC

WHAT IS NETWORK-LAYER CONFIDENTIALITY ?

Between two network entities:

- Sending entity encrypts datagram payload, payload could be:
  - TCP or UDP segment, ICMP message, OSPF message ....
- All data sent from one entity to other would be hidden:
  - Web pages, e-mail, P2P file transfers, TCP SYN packets ...
- "blanket coverage"

VIRTUAL PRIVATE NETWORKS (VPNS)



VIRTUAL PRIVATE NETWORKS (VPNS)

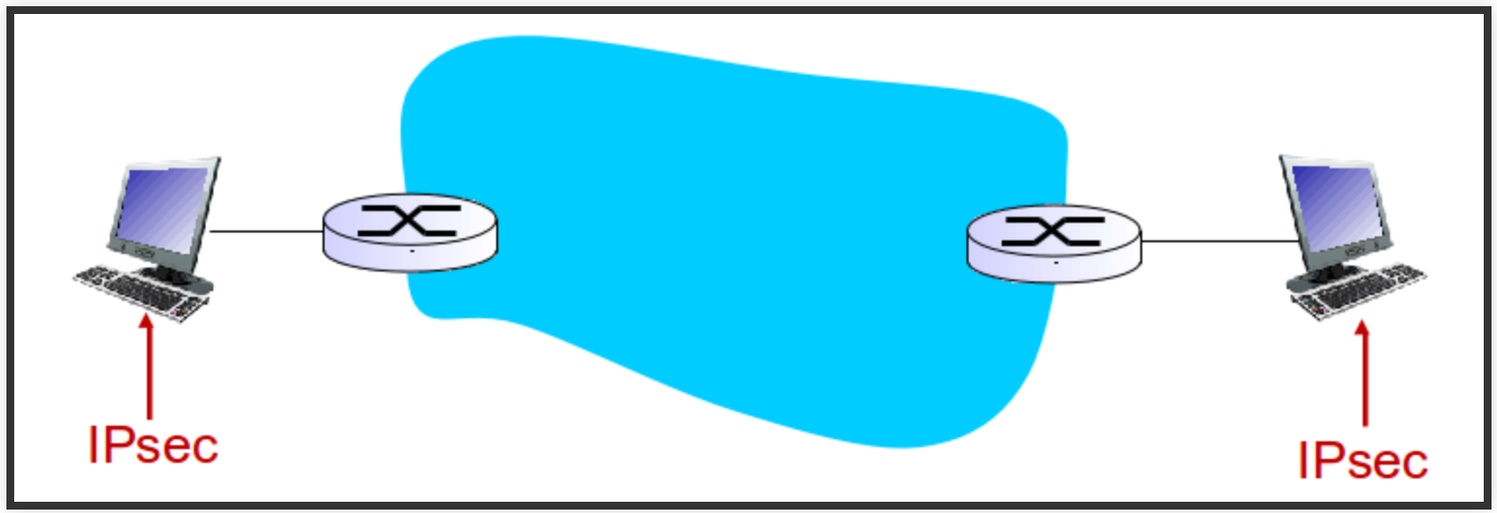
Motivation:

- Institutions often want private networks for security.
  - Costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet instead
  - Encrypted before entering public Internet
  - Logically separate from other traffic

IPSEC SERVICES

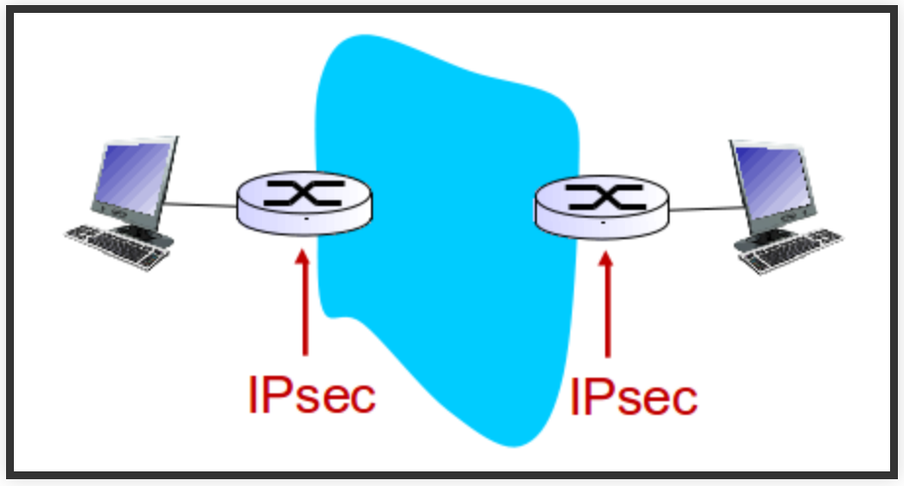
- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality
- Two protocols providing different service models:
  - AH - authentication header
  - ESP - encapsulating security payload

IPSEC TRANSPORT MODE



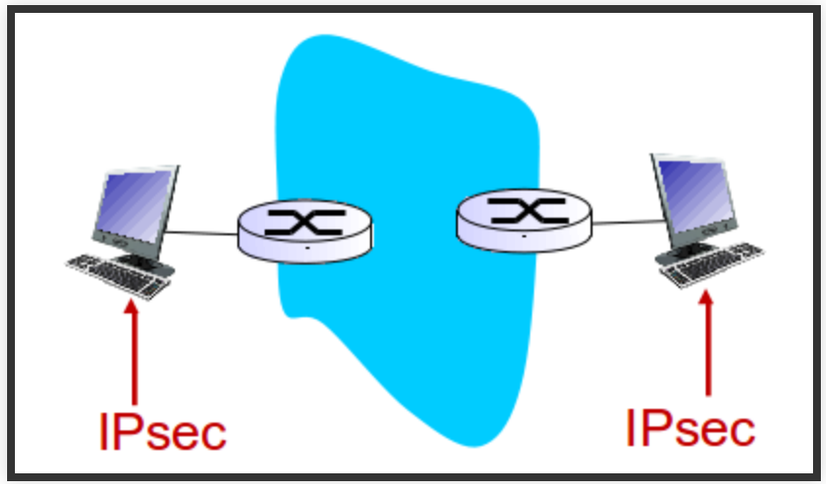
- IPsec datagram emitted and received by end-system
- Protects upper level protocols

IPSEC – TUNNELING MODE (1)



Edge routers IPsec-aware

IPSEC – TUNNELING MODE (2)



Hosts IPsec-aware

TWO IPSEC PROTOCOLS

Authentication Header (AH) protocol

- Provides source authentication & data integrity but not confidentiality

Encapsulation Security Protocol (ESP)

- Provides source authentication, data integrity, and confidentiality
- More widely used than AH

FOUR COMBINATIONS ARE POSSIBLE!

Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

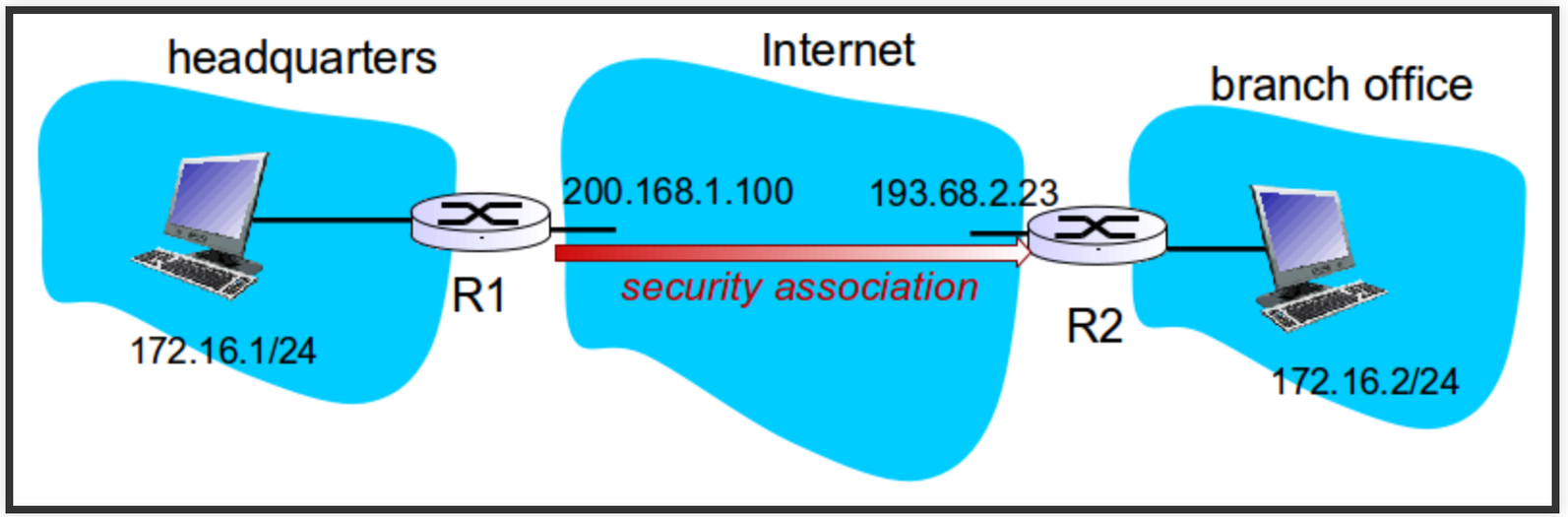
most common and most important

SECURITY ASSOCIATIONS (SAS)

SECURITY ASSOCIATIONS (SAS)

- Before sending data, “security association (SA)” established from sending to receiving entity
  - SAs are simplex: for only one direction
- Sending, receiving entitles maintain state information about SA
  - Recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!
- How many SAs in VPN w/ headquarters, branch office, and n traveling salespeople?

EXAMPLE SA FROM R1 TO R2



EXAMPLE SA FROM R1 TO R2

R1 stores for SA:

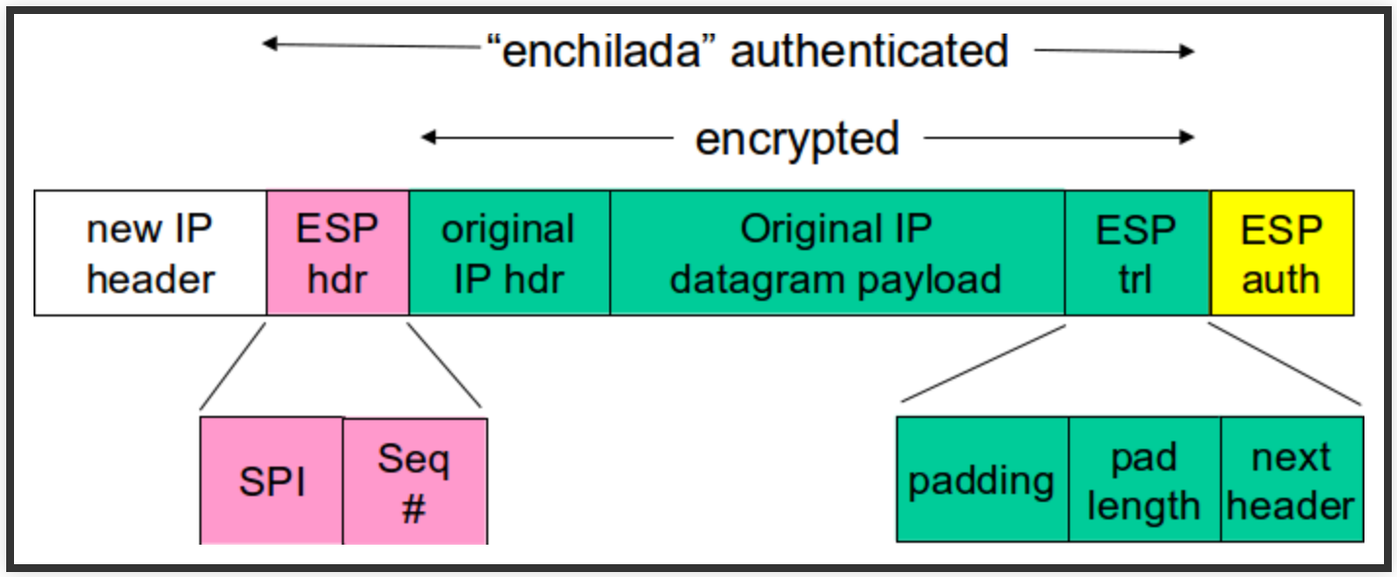
- 32-bit SA identifier: *Security Parameter Index (SPI)*
- Origin SA interface (200.168.1.100)
- Destination SA interface (193.68.2.23)
- Type of encryption used (e.g., 3DES with CBC)
- Encryption key
- Type of integrity check used (e.g., HMAC with MD5)
- Authentication key

SECURITY ASSOCIATION DATABASE (SAD)

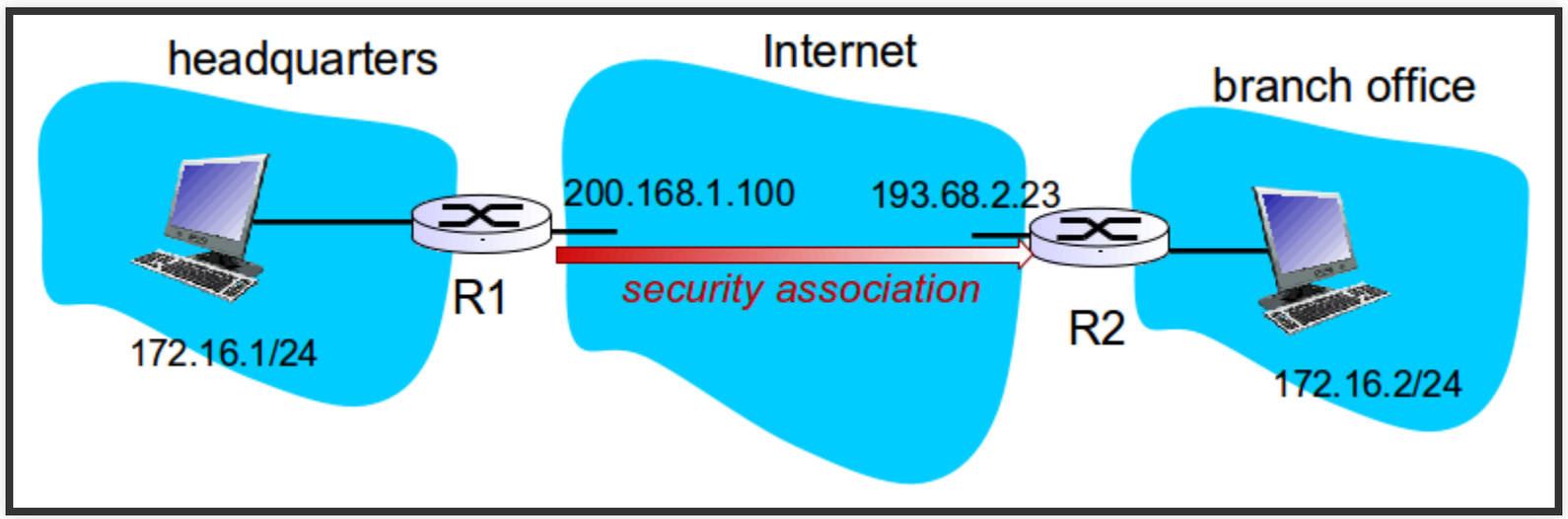
- Endpoint holds SA state in **security association database (SAD)**, where it can locate them during processing.
- With n salespersons, 2 + 2n SAs in R1's SAD
- When sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- When IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

IPSEC DATAGRAM

Focus for now on tunnel mode with ESP



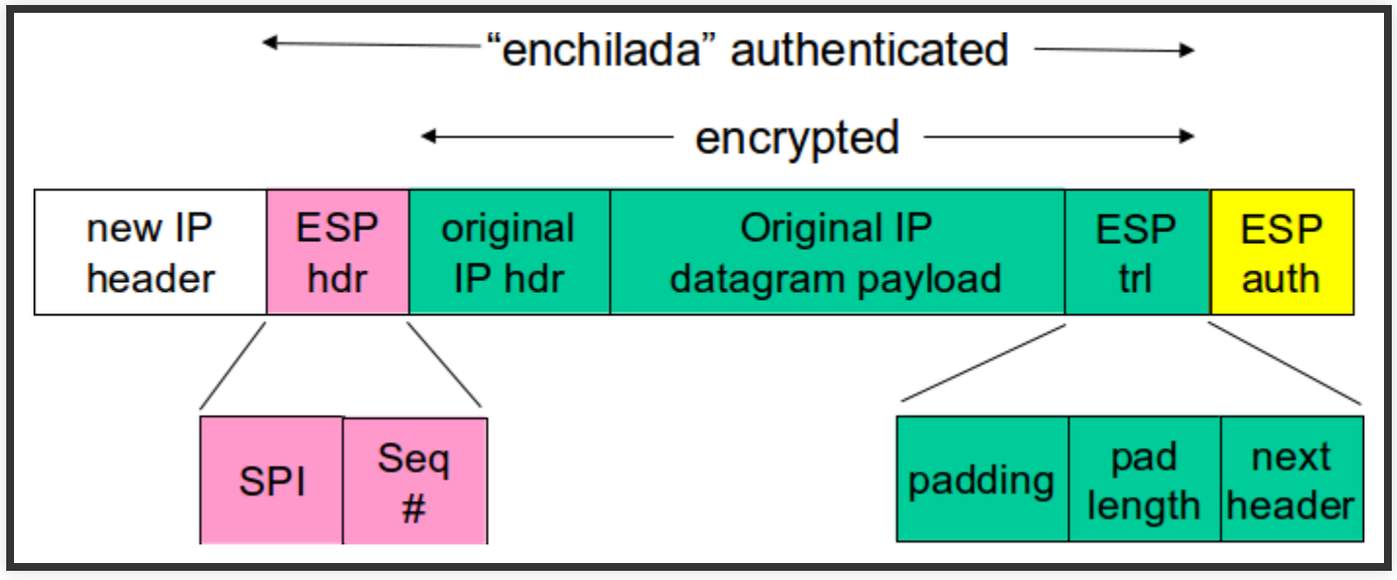
WHAT HAPPENS?



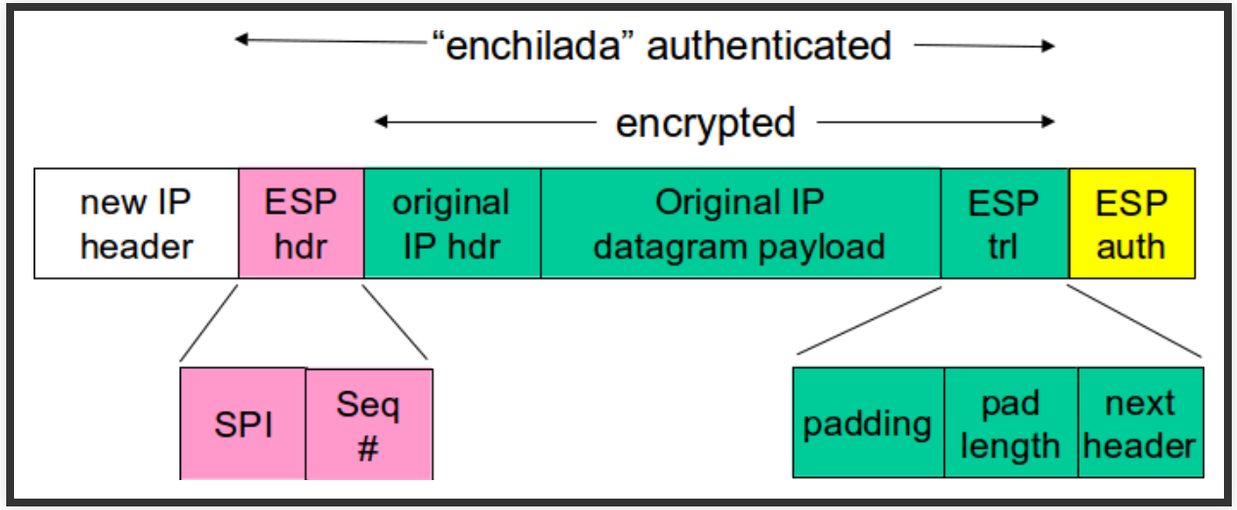
R1: CONVERT TO IPSEC DATAGRAM

- Appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- Encrypts result using algorithm and key specified by SA.
- Appends to front of this encrypted quantity the "ESP header", creating *enchilada*.
- Creates authentication MAC over the whole enchilada, using algorithm and key specified in SA
- Appends MAC to back of enchilada, forming payload;
- Creates new IP header, with all the classic IPv4 header fields; appends before payload.

R1: CONVERT ORIGINAL DATAGRAM TO IPSEC DATAGRAM



INSIDE THE PAYLOAD



- ESP trailer: Padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - Sequence number, to thwart replay attacks
- MAC in ESP auth field is created with shared secret key

IPSEC SEQUENCE NUMBERS

- For new SA, sender initializes seq. # to 0
- Each time datagram is sent on SA:
  - Sender increments seq # counter
  - Places value in seq # field

IPSEC SEQUENCE NUMBERS

Goal

- Prevent attacker from sniffing and replaying a packet
- Receipt of duplicate, authenticated IP packets may disrupt service

Method

- Destination checks for duplicates
- Doesn’t keep track of all received packets; instead uses a window

SECURITY POLICY DATABASE (SPD)

- **Policy:** For a given datagram, sending entity needs to know if it should use IPsec
- Needs also to know which SA to use
  - May use: source and destination IP address; protocol number
- Info in **SPD** indicates **what** to do with arriving datagram
- Info in **SAD** indicates **how** to do it

SUMMARY: IPSEC SERVICES

- Suppose Trudy sits somewhere between R1 and R2. she doesn't know the keys.
  - Will Trudy be able to see original contents of datagram? How about source, dest IP address, transport protocol, application port?
  - Flip bits without detection?
  - Masquerade as R1 using R1's IP address?
  - Replay a datagram?

IKE: INTERNET KEY EXCHANGE

https://tools.ietf.org/html/rfc7296.html

IKE: INTERNET KEY EXCHANGE

- Previous examples: manual establishment of IPsec SAs in IPsec endpoints:

Example SA

```
SPI: 12345
Source IP: 200.168.1.100
Dest IP: 193.68.2.23
Protocol: ESP
Encryption algorithm: 3DES-cbc
HMAC algorithm: MD5
Encryption key: 0x7aeaca…
HMAC key: 0xc0291f…
```

- Manual keying is impractical for VPN with 100s of endpoints
- Instead use **IPsec IKE (Internet Key Exchange)**

IKE: PSK AND PKI

IKE PHASES

- IKE has two phases
  - **Phase 1:** Establish bi-directional IKE SA
    - Note: IKE SA different from IPsec SA
      - Aka ISAKMP security association
  - **Phase 2:** ISAKMP is used to securely negotiate IPsec pair of SAs
- Phase 1 has two modes: aggressive mode and main mode
  - Aggressive mode uses fewer messages
  - Main mode provides identity protection and is more flexible

IPSEC SUMMARY

- IKE message exchange for algorithms, secret keys, SPI numbers
- Either AH or ESP protocol (or both)
  - AH provides integrity, source authentication
  - ESP protocol (with AH) additionally provides encryption
- IPsec peers can be two end systems, two routers/firewalls, or a router/firewall and an end system

QUESTIONS