Topics Covered in Course

**Attacks and hacker tools**
- Reconnaissance
- Network mapping
- Port scanning
- Sniffing
- IP address spoofing
- Session hijacking
- DoS

**Defenses**
- Firewalls
- Intrusion detection
- IP traceback
- Cryptography
  - **Secure protocols**
    - 802.11 security
    - PGP
    - SSL
    - IPsec (and VPN)

IEEE 802.11 Wireless LAN

- **802.11b**
  - 2.4-2.485 GHz unlicensed radio spectrum
  - up to 11 Mbps
  - Direct sequence spread spectrum (DSSS) in physical layer: all hosts use same chipping code

- **802.11a**
  - 5-6 GHz range
  - up to 54 Mbps
  - Physical layer: orthogonal frequency division multiplexing (OFDM)

802.11g

- 2.4-2.485 GHz range
- up to 54 Mbps
- OFDM
- All use CSMA/CA for multiple access
- All have base-station and ad-hoc versions
- All allow for reducing bit rate for longer range

802.11n

- 2.4-2.485 GHz range
- up to 54 Mbps
- OFDM
- All use CSMA/CA for multiple access
- All have base-station and ad-hoc versions
- All allow for reducing bit rate for longer range

802.11ac

- 5-6 GHz range
- up to 54 Mbps
- OFDM
- All use CSMA/CA for multiple access
- All have base-station and ad-hoc versions
- All allow for reducing bit rate for longer range

802.11ad

- 5-6 GHz range
- up to 54 Mbps
- OFDM
- All use CSMA/CA for multiple access
- All have base-station and ad-hoc versions
- All allow for reducing bit rate for longer range

802.11ax

- 5-6 GHz range
- up to 54 Mbps
- OFDM
- All use CSMA/CA for multiple access
- All have base-station and ad-hoc versions
- All allow for reducing bit rate for longer range

802.11ay

- 5-6 GHz range
- up to 54 Mbps
- OFDM
- All use CSMA/CA for multiple access
- All have base-station and ad-hoc versions
- All allow for reducing bit rate for longer range

IEEE 802.11 Architecture

- Each station and AP has its own MAC address

Channels and association

- Each AP is assigned SSID and a channel
- 11 overlapping channels: 1, 6, 11 are non-overlapping
- Each wireless station needs to associate with one AP:
  - Frames are sent to and received from associated AP

Beacon frames

- APs make presence known w/ beacons
- Wireless station chooses AP and initiates association protocol
  - Joins subnet to which AP belongs
  - After association, DHCP to get IP address, gateway, DNS server
### 802.11 Frame

Transfer from AP to host:
- Address 1 = host MAC
- Address 2 = AP address
- Address 3 = source address (eg router MAC)
- Address 4: N/A

Transfer from host to AP:
- Address 1 = AP address
- Address 2 = host MAC address
- Address 3 = dest address (eg router MAC)
- Address 4: N/A

- AP/host examines address 1 to determine if 802.11 frame is for it.
- More to come about other fields

---

### 802.11 security issues

**Eavesdropping**
- Trivial
- Identity in 802.11
- Non-malleable identify desirable
- MAC address of WAN card is easily changed
- Alternatives: passwords, public keys

**Access control lists**
- When based on MAC addresses:
  - Attacker sniffs channel, obtains valid MAC address
  - Attacker modifies its MAC address

**Firewalls**
- Firewalls restrict what flows in and out of enterprise network
- Wireless stations can bypass firewall
  - Thus wireless stations should be outside of firewall and treated as an external host

---

### WEP Overview

**Authentication**
- All authorized stations have same shared key

**Data integrity**
- Data received is the data sent

**Privacy**
- Symmetric encryption
Review: Symmetric Stream Ciphers

- Combine each byte of keystream with byte of plaintext to get ciphertext
- $m(i) =$ ith unit of message
- $ks(i) =$ ith unit of keystream
- $c(i) =$ ith unit of ciphertext
- $c(i) = ks(i) \oplus m(i)$ (\(\oplus\) = exclusive or)
- $m(i) = ks(i) \oplus c(i)$

Review: Problems with stream ciphers

Known plain-text attack
- There's often predictable and repetitive data in communication messages
- Attacker receives some ciphertext $c$ and correctly guesses corresponding plaintext $m$
- $ks = m \oplus c$
- Attacker now observes $c'$ created with same $ks$
  - Wants $m'$
  - $m' = ks \oplus c'$

Even easier
- Attacker obtains two ciphertexts, $m$ and $m'$, generating with same key sequence
- $c \oplus c' = m \oplus m'$
- There are well known methods for decrypting 2 plaintexts given their XOR

Summary: Never use same key sequence!

WEP encryption (1)

Per frame keystream
- Each side has 40-bit shared key
- Sender creates 24-bit initialization vector (IV), appends to key
- Appended 64-bit key inputted into pseudo random number generator to get keystream
- Data in frame + CRC is encrypted with keystream:
  - Bytes of keystream are XORed with bytes of data
  - IV is appended to encrypted data to create payload
  - Payload inserted into 802.11 frame

WEP encryption (2)

- WEP stream cipher: RC4
- IV goal: avoid repeating any portion of keystream
- WEP CRC is called Integrity Check Value (ICV)
**WEP encryption (3)**

- IV (per frame)
- \(K_S\): 40-bit secret symmetric
- Plaintext frame data plus CRC

New IV for each frame

**WEP encryption overview (4)**

- Receiver extracts IV
- Inputs IV and shared secret key into pseudo-random generator, gets keystream
- XORs keystream with encrypted data to decrypt data + CRC
- Verifies integrity of data with CRC

**WEP Authentication**

- Authentication request
- Nonce (128 bytes)
- Nonce encrypted shared key
- Success if decrypted value equals nonce

**Review End-point authentication 4.0**

- **Goal**: avoid playback attack
- **Nonce**: number (R) used only once-in-a-lifetime
- **ap4.0**: to prove Alice “live”, Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

“I am Alice”

Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!
Problem with WEP Authentication

Plaintext attack
- Attacker sniffs nonce, m, sent by AP
- Attacker sniffs response sent by station:
  - IV in clear
  - Encrypted nonce, c
- Attacker calculates keystream ks = m ⊕ c, which is the keystream for the IV.
- Attacker then requests access to channel, receives nonce m'
- Attacker forms response c' = ks ⊕ m' and IV
- Server decrypts, matches m' and declares attacker authenticated!

WEP Privacy

- Once station is authenticated, station and AP send encrypted data to each other
- New IV for each frame

Problem with WEP privacy

Security hole:
- 24-bit IV gives 16 million IVs: can see 16 million frames in one day
- one IV per frame -> IV's eventually reused
- IV transmitted in plaintext -> IV reuse detected
- Attack:
  - Trudy guesses some of Alice's plaintext d₁ d₂ d₃ d₄ ...
  - Trudy sniffs: cᵢ = dᵢ ⊕ kᵢIV
  - Trudy computes keystream kᵢIV =cᵢ ⊕ dᵢ
  - Trudy knows encrypting keystream k₁IV k₂IV k₃IV ...
  - Next time IV is used, Trudy can decrypt!

Summary of WEP flaws

One common shared key
- If any device is stolen or compromised, must change shared key in all devices
- No key distribution mechanism
- Infeasible for large organization: approach doesn't scale

Crypto is flawed
- Early 2001: Integrity and authentication attacks published
- August 2001 (FMS attack): can deduce RC4 key after observing several million packets
- AirSnort application allows casual user to decrypt WEP traffic

Crypto problems
- 24-bit IV too short
- CRC (ICV) flawed, does not prevent adversarial modification of intercepted packets
- Cryptanalytic attack allows eavesdroppers to learn key after observing several millions of packets
IEEE 802.11i

- Much stronger encryption
- Extensible set of authentication mechanisms
  - Employs 802.1X authentication
  - For our purposes = Extensible Authentication Protocol (EAP)
- Key distribution mechanism
  - Typically public key cryptography

802.1X Authentication

3 components
- Software in wireless station
- AP: gatekeeper, allowing access only after authentication
- Authentication server e.g. RADIUS

AP is separated from authentication server
- Allows one server to handle many APs
- Challenge type up to vendor
- Secret info is not sent over air in plaintext

1. Client makes an association with AP
2. AP places client in an unauthenticated holding area; AP sends an authentication request to client
3. Client sends user ID to AP, which forwards it to server
4. Server sends challenge via AP to client
5. Client responds to challenge
6. Server verifies response, provides fresh session keys

802.11 security summary

- SSID and access control lists provide minimal security
  - no encryption
- WEP provides encryption, but is easily broken
- New emerging protocol: 802.11i
  - Back-end authentication server
  - Public-key cryptography for authentication and master key distribution
  - Strong symmetric crypto techniques
Topics Covered in Course

**Attacks and hacker tools**
- Reconnaissance
- Network mapping
- Port scanning
- Sniffing
- IP address spoofing
- Session hijacking
- DoS

**Defenses**
- Firewalls
- Intrusion detection
- IP traceback
- Cryptography

**Secure protocols**
- 802.11 security
- PGP
- SSL
- IPsec (and VPN)

Secure e-mail: PGP

- PGP = Pretty Good Privacy
- It is available free on a variety of platforms.
- inventor, Phil Zimmerman, was target of 3-year federal investigation.
- Based on well known algorithms.
- Not developed or controlled by governmental or standards organizations

Secure e-mail: PGP

- Consist of five services:
  - Authentication
  - Confidentiality
  - Compression
  - E-mail compatibility
  - Segmentation

**Secure e-mail: Authentication and Message Integrity**

- Alice wants to provide sender authentication message integrity.

- Alice digitally signs message.
- Sends both message (in the clear) and digital signature.
Authentication: PGP

- Hash:
  - SHA-1
  - 160 bits
- Public key cryptography:
  - RSA

Secure e-mail: Confidentiality

- Alice wants to send confidential e-mail, m, to Bob.
  - Alice generates random symmetric key, KS.
  - Encrypts message with KS (for efficiency)
  - Also encrypts KS with Bob’s public key.
  - Sends both KS(m) and KB(KS) to Bob.

Confidentiality: PGP

- Session key: 128 bits
- Symmetric encryption:
  - CAST-128 or IDEA or 3DES
- Public key encryption
  - RSA
Secure e-mail: Confidentiality and Authentication

- Alice wants to provide secrecy, sender authentication, message integrity.

\[ m \rightarrow H() \rightarrow KA(H(m)) \rightarrow KS + KB(KS) \rightarrow K_B \rightarrow \text{Internet} \]

Alice uses three keys: her private key, Bob’s public key, newly created symmetric key

Format of PGP Message

- Users may have multiple key pairs
  - Key ID’s: last 64 bits of public key
- Message component
  - Signature component
    - Timestamp
    - Key ID of \( K_A \)
    - Mess digest: \( K_A(H(m)) \)
- Session key component
  - key ID of \( K_B \)
  - Session key: \( K_S \)

PGP key rings

- Each node has two key rings:
  - Public/private key pairs owned by that node
  - Public/private key pairs of other users
- For the keys of other users, for each key track:
  - timestamp
  - key ID
  - public key
  - user id: e-mail address
  - key legitimacy
  - signatures

PGP Public Key Management

- How does Alice know that the recipient’s public key is really Bob’s public key?

  **Approaches:**
  1. Physically getting key from Bob
  2. Verify key on telephone
  3. Certification authority
  4. Web of trust

  **For each key on ring:**
  - key legitimacy field
  - indicates the trust of the validity of the PK
  - signatures for PK
  - for each signature, signature trust field
Public key management: example

- Suppose Alice inserts new public key on key ring. If Alice is owner, trust is ultimate.
- Otherwise, Alice must assign trust value to owner of key:
  - unknown
  - untrusted
  - marginally trusted
  - completely trusted.
- New public key may come with signatures. For each signature, PGP searches ring to see if author of signature is owner of key. Sigtrust value = trust value of owner.
- Key legitimacy = ultimate if one signature is ultimate. Otherwise, determined from formula based on sigtrust values; above threshold, key is considered legit.

Example

Revoking Public Keys

- The owner issue a key revocation certificate.
- Normal signature certificate with a revoke indicator.
- Corresponding private key is used to sign the certificate.

PGP summary

- PGP provides security at the application layer to a single application
- Provides:
  - Authentication, integrity, confidentiality
- Public key verification
  - Web of trust
**Topics Covered in Course**

**Attacks and hacker tools**
- Reconnaissance
- Network mapping
- Port scanning
- Sniffing
- IP address spoofing
- Session hijacking
- DoS

**Defenses**
- Firewalls
- Intrusion detection
- IP traceback
- Cryptography

**Secure protocols**
- 802.11 security
- PGP
- SSL
- IPsec (and VPN)

**SSL: Secure Sockets Layer**

- Most widely deployed security protocol
  - Supported by almost all browsers and web servers
  - HTTPS
  - Tens of billions $ spent per year over SSL

- Originally designed by Netscape in 1993
- Number of variations:
  - TLS: transport layer security, RFC 2246

- Provides
  - Confidentiality
  - Integrity
  - Authentication
- Original goals:
  - Had Web e-commerce transactions in mind
  - 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**

- Normal Application
- Application with SSL

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

**Could do something like PGP:**

- But want to send byte streams & interactive data
- Want a set of secret keys for the entire connection
- Want certificate exchange part of protocol: handshake phase
Toy SSL: a simple secure channel

- **Handshake**: Alice and Bob use their certificates and 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 a series of records.
- **Connection Closure**: Special messages to securely close connection.

Toy: A simple handshake

- **Handshake**:
  - Hello
  - Certificate
  - \( K_B(MS) = EMS \)

- **Purpose**: Message integrity
- **No encryption!**
- **Notation**: \( MD_m = H(s||m) \); send \( m||MD_m \)

Toy: Key derivation

- **Considered bad to use same key for more than one cryptographic operation**
  - Use different keys for message authentication code (MAC) and encryption.
- **Four keys**:
  - \( K_c \) = encryption key for data sent from client to server.
  - \( M_c \) = MAC key for data sent from client to server.
  - \( E_s \) = encryption key for data sent from server to client.
  - \( M_s \) = 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.
**Toy: Data 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.
- Instead, break stream in series of records
  - Each record carries a MAC
  - Receiver can act on each record as it arrives
- Issue: in record, receiver needs to distinguish MAC from data
  - Want to use variable-length records

<table>
<thead>
<tr>
<th>length</th>
<th>data</th>
<th>MAC</th>
</tr>
</thead>
</table>

**Toy: Sequence Numbers**
- Attacker can capture and resend record or re-order records
- Solution: put sequence number into MAC:
  - $MAC = MAC(M_x, \text{sequence}||\text{data})$
  - Sequence number serves a nonce for record
- Attacker could still replay all of the records
  - Use session nonce as well

**Toy SSL: summary**
- Hello, certificate, nonce
- $K_{PS}(MS) = EMS$
- Type 0, seq 1, data
- Type 0, seq 2, data
- Type 0, seq 1, data
- Type 0, seq 3, data
- Type 1, seq 4, close
- Type 1, seq 2, close
Toy SSL isn't complete

- How long are the fields?
- What encryption protocols?
- No negotiation
  - Allow client and server to support different encryption algorithms
  - Allow client and server to choose together specific algorithm before data transfer

Most common symmetric ciphers in SSL

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

Public key encryption
- RSA

SSL Cipher Suite

- Cipher Suite
  - Public-key algorithm
  - Symmetric encryption algorithm
  - MAC (hash) algorithm
- SSL supports a variety of cipher suites
- Negotiation: client and server must agree on cipher suite
- Client offers choice; server picks one

Real SSL: Handshake (1)

Purpose
1. Negotiation: agree on crypto algorithms
2. Establish keys
3. Client authentication (optional)
Real SSL: Handshake (2)

1. Client sends list of algorithms it supports, along with client nonce
2. Server chooses algorithms from list; sends back: choice + certificate + server nonce
3. Client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. Client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. Client sends a MAC of all the handshake messages
6. 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 the stronger algorithms from list
- Last 2 steps prevent this
- Last two messages are encrypted

SSL Record Protocol

- data
- data fragment
- MAC
- record header
- encrypted data and MAC

record header: content type; version; length
MAC: includes sequence number, MAC key M_x
Fragment: each fragment 2^{14} bytes

Content types in record header

- application_data (23)
- alert (21)
  - signaling errors during handshake
  - signal connection closure
- handshake (22)
  - initial handshake messages are carried in records of type "handshake"
- change_cipher_spec (20)
  - indicates change in encryption and authentication algorithms
SSL Record Format

<table>
<thead>
<tr>
<th>content type</th>
<th>SSL version</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data and MAC encrypted (symmetric algo)

Comments about trace messages

- **ClientHello**
  - random: 32-byte nonce
- **ServerHello**
  - cipher suite: RSA key exchange, DES-CBC message encryption, SHA digest
  - random: 32-byte nonce
  - session_id: used for session resumption

- **Certificate**
  - X.509 format
  - Subject: company info
  - Issuer: CA
  - certificate = public key
- **ClientKeyExchange**
  - Includes encrypted PreMasterSecret
- **Finished**
  - MAC of concatenation of handshake messages

Key derivation

- Client random, server random, and pre-master secret input into pseudo random-number generator.
  - Produces master secret
- Master secret, client and server random numbers into another random-number generator
  - Produces "key block"
- 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)
SSL Performance

- Big-number operations in public-key crypto are CPU intensive
- Server handshake
  - Typically over half SSL handshake CPU time goes to RSA decryption of the encrypted pre_master_secret
- Client shake
  - Public key encryption is less expensive
  - Server is handshake bottleneck
- Data transfer
  - Symmetric encryption
  - MAC calculation
  - Neither as CPU intensive as public-key decryption

Session resumption

- Full handshake is expensive: CPU time and number of RRT
- If the client and server have already communicated once, they can skip handshake and proceed directly to data transfer
  - For a given session, client and server store session_id, master_secret, negotiated ciphers
- Client sends session_id in ClientHello
- Server then agrees to resume in ServerHello
  - New key_block computed from master_secret and client and server random numbers

Client authentication

- SSL can also authenticate client
- Server sends a CertificateRequest message to client

Topics Covered in Course

- Attacks and hacker tools
  - Reconnaissance
  - Network mapping
  - Port scanning
  - Sniffing
  - IP address spoofing
  - Session hijacking
  - DoS

- Defenses
  - Firewalls
  - Intrusion detection
  - IP traceback
  - Cryptography
    - Secure protocols
    - 802.11 security
    - PGP
    - SSL
    - IPSec (and VPN)
Security at different layers

- Application layer: PGP
- Transport layer: SSL
- Network layer: IPsec

IPsec approach:

<table>
<thead>
<tr>
<th>HTTP/SMTP/IM</th>
<th>TCP/UDP/ICMP</th>
<th>IPsec</th>
</tr>
</thead>
</table>

IP Security

- IP datagrams have no inherent security
  - IP source address can be spoofed
  - Content of IP datagrams can be sniffed
  - Content of IP datagrams can be modified
  - Old IP datagrams can be replayed

IPSec is a method for protecting IP datagrams
- Standardized by IETF: dozens of RFCs.
- Only sender and receiver have to be IPsec compliant
  - Rest of network can be regular IP

Applications of IPsec

- Secure branch office connectivity over Internet
  - No need for private network: save money
- Secure remote access over Internet
  - End user's PC equipped with IPsec gets secure access to company network
  - Reduces cost of toll charges; permits high-speed access
- Secure interconnection among corporate partners
  - Ensuring authentication and confidentiality

IP Security Scenario
**IPsec services**
- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality

**IPsec Transport Mode**
- IPsec datagram emitted and received by end-system.
- Protects upper level protocols

**IPsec - tunneling mode**
- End routers are IPsec aware. Hosts need not be.

**IPsec: Overview**

*Main protocols:*
- **Authentication Header (AH):**
  - Provides integrity and authenticity
- **Encapsulated Security Payload (ESP):**
  - Encrypts data and even headers
- **Internet Key Exchange (IKE) Protocol:**
  - Allows peers (end systems, routers, firewalls) to agree on methods, algorithms, keys
IPsec services summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>AH</th>
<th>ESP (encryption only)</th>
<th>ESP (encryption plus authentication)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access control</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Connectionless integrity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data origin authentication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rejection of replayed packets</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Confidentiality</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Limited traffic flow confidentiality</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Security Associations (SAs)

- Flows of IPsec data are organized into SAs
- An SA is one way from sender peer to receiver peer

SA uniquely defined by:
- Security parameter index (SPI): 32 bits
- IP destination address
- Security protocol identifier (identifies AH or ESP)

- An SA times out after configured number of seconds or bytes

SA selectors

- IPsec source peer doesn't necessarily apply IPsec to all outgoing packets:
  - Each SA is associated with "selectors":
    - Source, destination IP addresses
    - Transport layer protocol
    - Source, destination port numbers
    - IPv4 type of service field
    - IPv6 fields
- When datagram arrives to IPsec source peer, peer examines selectors, finds SA, and applies appropriate IPsec processing:
  - Nothing, AH, ESP, ESP with AH
- Security Policy Database (SPD)
  - Contains mappings from selectors to SAs

AH protocol

- Support for data integrity, source authentication, and replay attacks
  - But not encryption
- Authentication uses a MAC with a shared secret key.
- Source peer first establishes an SA with destination peer using IKE protocol
- AH protocol then processes IP datagrams that belong to the SA
AH protocol framing

Before applying AH

Transport mode

Tunnel mode

In each case, the protocol field in the left-most IP header is 51, indicating that the AH protocol is used.

Question: How does destination peer determine upper-layer protocol to which it passes the payload?

Authentication Header

- Next header: indicates if datagram carries TCP, UDP, ICMP, etc.
- Payload length: length of authentication header
- SPI: identifies SA
- Sequence number: incremented for each datagram in SA
- Authentication data: contains MAC for this packet

AH: 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
- Goal:
  - Prevent attacker from sniffing and replaying a packet
  - Receipt of duplicate, authenticated IP packets may disrupt service
- Method:
  - Destination checks for duplicates
  - But doesn’t keep track of ALL received packets; instead uses a window

Algorithm at receiver

N is highest sequence # rcvd.

Default W=64

1. If rcvd packet falls in window, packet is new, and MAC checks ➔ slot in window marked
2. If rcvd packet is to right of window, new, MAC checks ➔ window advanced & right-most slot marked
3. If rcvd packet is left of window, or already marked, or fails MAC check ➔ packet is discarded
**AH: MAC**
- MD5-96 or SHA-1-96
- Calculated over:
  - IP header fields that do not change in transit (not TTL or checksum fields)
  - + AH header excluding Authentication Data field (the MAC itself)
  - + entire upper-layer payload (e.g., TCP segment or inner IP datagram in tunnel mode)

**AH: transport mode**

Before applying AH

<table>
<thead>
<tr>
<th>Original IP header</th>
<th>TCP header</th>
<th>Data</th>
</tr>
</thead>
</table>

Transport mode

<table>
<thead>
<tr>
<th>Original IP</th>
<th>AH header</th>
<th>TCP header</th>
<th>Data</th>
</tr>
</thead>
</table>

Authentication covers the entire original IP datagram excluding mutable IP fields.

Source IP address equals source host address;
Destination IP address = destination host address

**AH: tunnel mode**

Before applying AH

<table>
<thead>
<tr>
<th>Original IP header</th>
<th>TCP header</th>
<th>Data</th>
</tr>
</thead>
</table>

Tunnel mode

<table>
<thead>
<tr>
<th>New IP header</th>
<th>AH header</th>
<th>Original IP header</th>
<th>TCP header</th>
<th>Data</th>
</tr>
</thead>
</table>

In tunnel mode, the entire original datagram is authenticated:

- Original datagram is carried in the payload of the new datagram - fields do not change in payload.
- If both peers are routers, then source and dest IP addresses are those of the routers.

**ESP protocol**

- ESP protocol provides confidentiality through encryption
- Optionally provides same authentication services as the AH protocol
- Symmetric encryption with shared secret key.
- Source peer first establishes an SA with destination peer using IKE protocol
- ESP protocol then processes IP datagrams that belong to the SA
**Transport mode**

Attacker can perform traffic analysis: examine protocol numbers, ultimate destination, and other IP fields.

**Tunnel mode**

Can counter traffic analysis. Even ultimate destination is hidden.

Hosts do not have to do encryption - instead done by routers and firewalls.

When in routers/firewalls, key distribution is easier.

**ESP header/trailer**

Principal purpose of padding: encryption algorithm may require input to be multiple of block size.

**ESP example**

1. Remote host prepares inner datagram with dest. address of server.
2. ESP trailer appended to datagram; result encrypted.
3. ESP header added; MAC across all of it appended
4. Datagram header added. SA = remote host, DA = firewall
5. Firewall receives datagram and MAC verified; using SPI in ESP header, firewall decrypts to get plaintext inner datagram. Firewall sends inner datagram to server.
Padding

- Padding field serves several purposes:
  - Encryption algorithm may require input to be multiple of block size
  - Pad length and next header fields need

Possible encryption algorithms

- DES
- 3DES
- AES
- RC5
- IDEA
- 3-IDEA
- CAST
- Blowfish
- ...

Key Management (IKE)

Two broad approaches

- Manual
  - Systems administrator manually configures each end system, IPsec router/firewall with its own secret symmetric keys.

- Automated
  - On-demand creation secret symmetric keys
  - As usual, use public-key cryptography to distribute symmetric keys
  - Default protocol is ISAKMP/Oakley
  - ISAKMP = Internet Security Association and Key Management Protocol

Diffie-Hellman public-key encryption

- Prior (non-secret) agreement between peers A and B on two global parameters:
  - Large prime number q
  - Integer a (primitive root of q)
- A and B independently select random private keys: $X_A$ and $X_B$ (integers)
- A sends B its public key $Y_A = a^{X_A} \mod q$
- B sends A its public key $Y_B = a^{X_B} \mod q$
- Each side computes same session key:
  - $K = (Y_B)^{X_A} \mod q = (Y_A)^{X_B} \mod q = a^{X_A X_B} \mod q$
**Oakley: Refinement of D-H**

- Cookies
  - D-H is computationally expensive and thus vulnerable to clogging DoS attack
- Uses nonces to protect against replay attacks
- Authenticates D-H exchange to thwart man-in-the-middle attack
  - For example, exchanging certificates

**Oakley: Cookie Exchange**

- Attacker, spoofing multiple source addresses, gets victim to do numerous D-H exponentiations, clogging victim
- Idea: first perform cookie exchange before exponentiating
  - Idea similar to SYN flood cookies (see DoS lecture)
- A sends B some random data and B returns it; similarly B sends A random data and A returns it
  - If ack comes back, perform exponentiation

**ISAKMP**

- Defines procedures & packet formats to establish security associations (SAs).
- Defines payloads for exchanging key generation material and authentication data.
- Consistent format independent of specific key exchange protocol (e.g. Oakley), symmetric key algorithm, authentication mechanism

**ISKMP message**

- Consists of header followed by one or more payloads
- Typically sent over UDP
**ISAKMP header**

- Next payload: type of first payload in message
- Length: total length of message (header + payloads)

**Payloads**

- Payload header:

**Some payload types**
- SA payload: begins the establishment of an SA
- Proposal payload: contains information used during SA negotiation
- Key exchange payload: e.g., for Oakley/D-H key exchange
- Certificate payload
- Nonce (cookies) payload

**ISAKMP message exchanges: example “base exchange”**

1. I to R: SA; Nonce begin negotiation
2. R to I: SA; Nonce SA agreed on
3. I to R: key; ID_I key + initiator id
4. R to I: key; ID_R key + responder id

I = initiator; R = responder

**Summary of IPsec**

- I and R exchange ISAKMP messages to define SA, algorithms, secret keys
- The SA uses either the AH or the ESP protocol
- The AH protocol provides integrity and source identification
- The 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