Chapter 10: Cipher Techniques

- Problems
  - What can go wrong if you naively use ciphers
- Cipher types
  - Stream or block ciphers?
- Networks
  - Link vs end-to-end use
- Examples
  - Security at the Network Layer (IPsec)

Problems

- Using cipher requires knowledge of *environment*, and *threats* in the environment, in which cipher will be used
  - Is the set of possible messages small?
  - Do the messages exhibit *regularities* that remain after encipherment?
  - Can an active wiretapper *rearrange* or *change* parts of the message?

Attack #1: Precomputation

- Set of possible messages M small
- Public key cipher f used
- Idea: precompute set of possible ciphertexts $f(M)$, build table $(m, f(m))$
- When ciphertext $f(m)$ appears, use table to find m
- Also called *forward searches*

Example

- Cathy knows Alice will send Bob one of two messages: enciphered BUY, or enciphered SELL
- Using public key e_{Bob}, Cathy precomputes m_1 = { BUY } e_{Bob}, m_2 = { SELL } e_{Bob}
- Cathy sees Alice send Bob m_2
- Cathy knows Alice sent SELL

Re-Ordered Blocks

- Alice sends Bob message
  - $n_{Bob} = 77$, $e_{Bob} = 17$, $d_{Bob} = 53$
  - Message is LIVE (11 08 21 04)
  - Enciphered message is 44 57 21 16
- Eve intercepts it, rearranges blocks
  - Now enciphered message is 16 21 57 44
- Bob gets enciphered message, deciphers it
  - He sees EVIL

Notes

- Digitally signing each block won't stop this attack
- Two approaches:
  - Cryptographically hash the *entire* message and sign it
  - Place sequence numbers in each block of message, so recipient can tell intended order
    - Then you sign each block

Statistical Regularities

- If plaintext repeats, ciphertext may repeat too
- Example using DES:
  - input (in hex):

    3231 3433 3635 3837 3231 3433 3635 3837
  - corresponding output (in hex):

    ef7c 4bb2 b4ce 6f3b ef7c 4bb2 b4ce 6f3b
- Fix: cascade blocks together (chaining)

What These Mean

- Use of strong cryptosystems, well-chosen (or random) keys not enough to be secure
- Other factors to consider:
  - Protocols directing use of cryptosystems
  - Ancillary information added by protocols
  - Implementation (not discussed here)
  - Maintenance and operation (not discussed here)

Stream, Block Ciphers

- E encryption function
  - $E_k(b)$ encryption of message b with key k
  - In what follows, $m = b_1 b_2$ …, each b_i of fixed length
- Block cipher
  - $E_k(m) = E_k(b_1)E_k(b_2)$ …
  - Example: DES (b_i = 64 bits, k = 56 bits)
- Stream cipher
  - $k = k_1 k_2$ …
  - $E_k(m) = E_{k1}(b_1)E_{k2}(b_2)$ …
  - If $k_1 k_2$ … repeats itself, cipher is *periodic* and the length of its period is one cycle of $k_1 k_2$ …
  - Will see several examples

Stream Ciphers

- Often (try to) implement one-time pad by xor'ing each bit of key with one bit of message
  - Example:

$$m = 00101\ldots$$
$$k = \ 10010\ldots$$
$$c = \ 10111\ldots$$

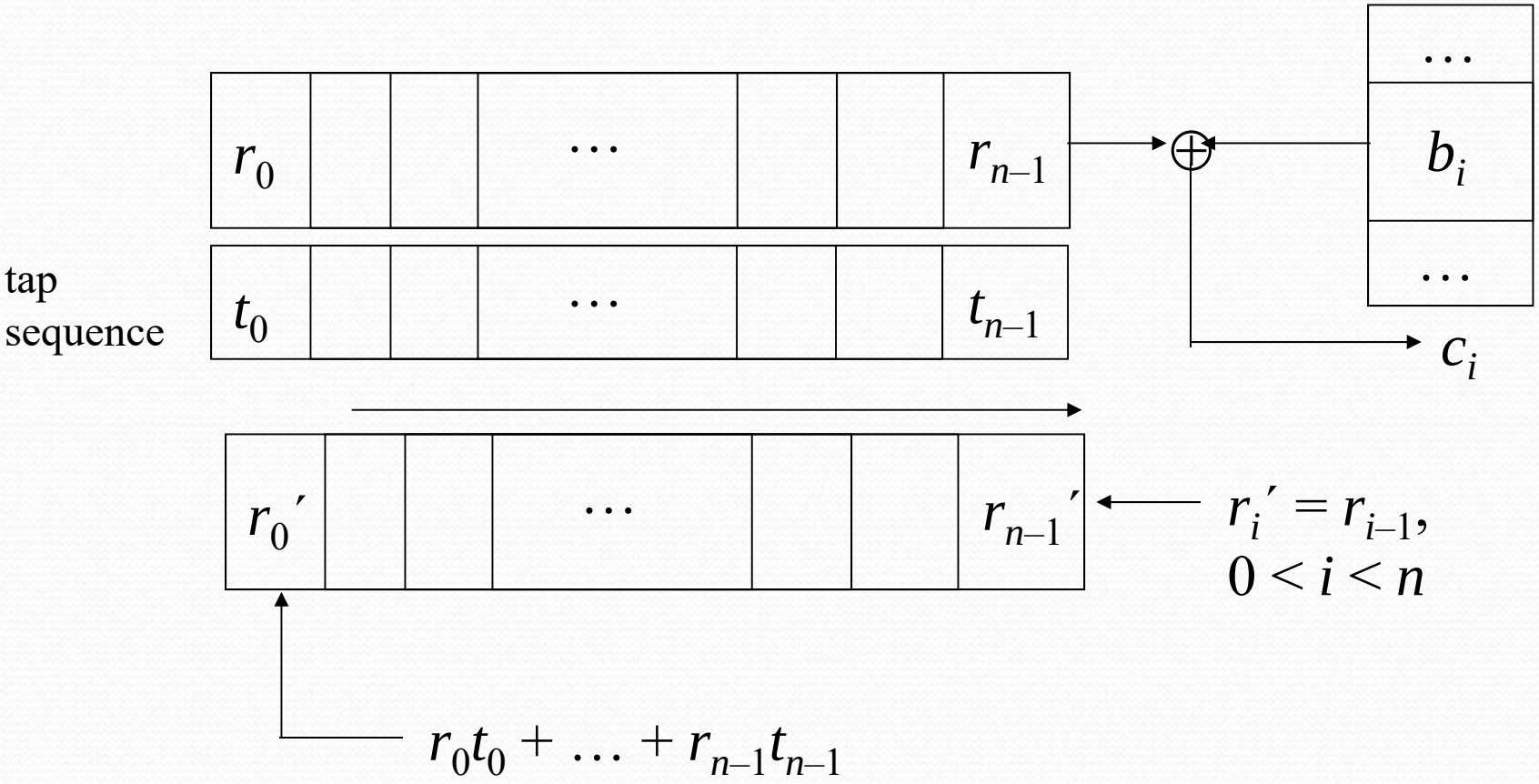
- But how to generate a good key?

Synchronous Stream Ciphers

- n-stage Linear Feedback Shift Register: consists of
  - n bit register $r = r_0 \ldots r_{n-1}$
  - n bit tap sequence $t = t_0 \ldots t_{n-1}$
  - Use:
    - Use r_{n-1} as key bit
    - Compute $x = r_0 t_0 \oplus \ldots \oplus r_{n-1} t_{n-1}$
    - Shift r one bit to right, dropping r_{n-1}, x becomes r_0

Acknowledgement: Matt Bishop

Operation


r_0
$\cdots$
r_{n-1}
t_0
$\cdots$
t_{n-1}
tap sequence
$\cdots$
b_i
$\cdots$
c_i
r_0'
$\cdots$
r_{n-1}'
$r_i' = r_{i-1}$, $0 < i < n$
$r_0t_0 + \ldots + r_{n-1}t_{n-1}$

Example

- 4-stage LFSR; t = 1001

r	k_i	*new bit computation*	*new r*
0010	0	01⊕00⊕10⊕01 = 0	0001
0001	1	01⊕00⊕00⊕11 = 1	1000
1000	0	11⊕00⊕00⊕01 = 1	1100
1100	0	11⊕10⊕00⊕01 = 1	1110
1110	0	11⊕10⊕10⊕01 = 1	1111
1111	1	11⊕10⊕10⊕11 = 0	0111
1110	0	11⊕10⊕10⊕11 = 1	1011

- Key sequence has period of 15 (010000111010110)

NLFSR

- n-stage Non-Linear Feedback Shift Register: consists of
  - n bit register $r = r_0 \ldots r_{n-1}$
  - Use:
    - Use r_{n-1} as key bit
    - Compute $x = f(r_0, \ldots, r_{n-1})$; f is any function
    - Shift r one bit to right, dropping r_{n-1}, x becomes r_0

    Note same operation as LFSR but more general bit replacement function

Example

- 4-stage NLFSR; $f(r_0, r_1, r_2, r_3) = (r_0 \,\&\, r_2) \mid r_3$

r	k_i	*new bit computation*	*new r*
1100	0	(1 & 0) | 0 = 0	0110
0110	0	(0 & 1) | 0 = 0	0011
0011	1	(0 & 1) | 1 = 1	1001
1001	1	(1 & 0) | 1 = 1	1100
1100	0	(1 & 0) | 0 = 0	0110
0110	0	(0 & 1) | 0 = 0	0011
0011	1	(0 & 1) | 1 = 1	1001

  - Key sequence has period of 4 (0011)

Eliminating Linearity

- NLFSRs not common
  - No body of theory about how to design them to have long period
- Alternate approach: *output feedback mode*
  - For E encipherment function, k key, r register:
    - Compute $r' = E_k(r)$; key bit is rightmost bit of r'
    - Set r to r' and iterate, repeatedly enciphering register and extracting key bits, until message enciphered
  - Variant: use a counter that is incremented for each encipherment rather than a register
    - Take rightmost bit of $E_k(i)$, where i is number of encipherment

Block Ciphers

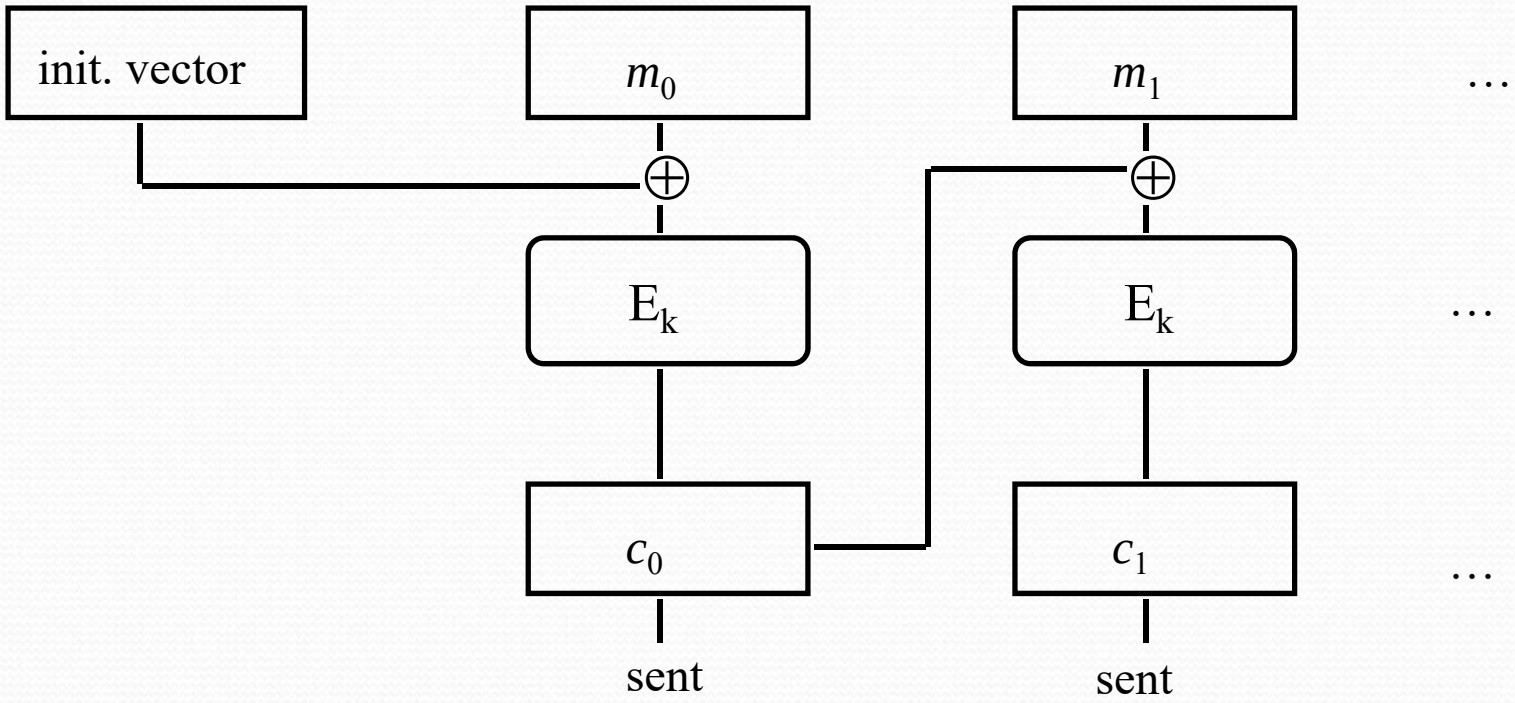
- Encipher, decipher multiple bits at once
- Each block enciphered independently
- Problem: identical plaintext blocks produce identical ciphertext blocks
  - Example: two database records
    - `MEMBER: HOLLY INCOME $100,000`
    - `MEMBER: HEIDI INCOME $100,000`
  - Encipherment:
    - `ABCQZRME GHQMRSIB CTXUVYSS RMGRPFQN`
    - `ABCQZRME ORMPABRZ CTXUVYSS RMGRPFQN`

Solutions

- Insert information about block's **position** into the plaintext block, then encipher
- *Cipher block chaining*:
  - Exclusive-or current plaintext block with previous ciphertext block:
    - $c_0 = E_k(m_0 \oplus I)$
    - $c_i = E_k(m_i \oplus c_{i-1})$ for i > 0

    where I is the initialization vector

CBC Mode Encryption

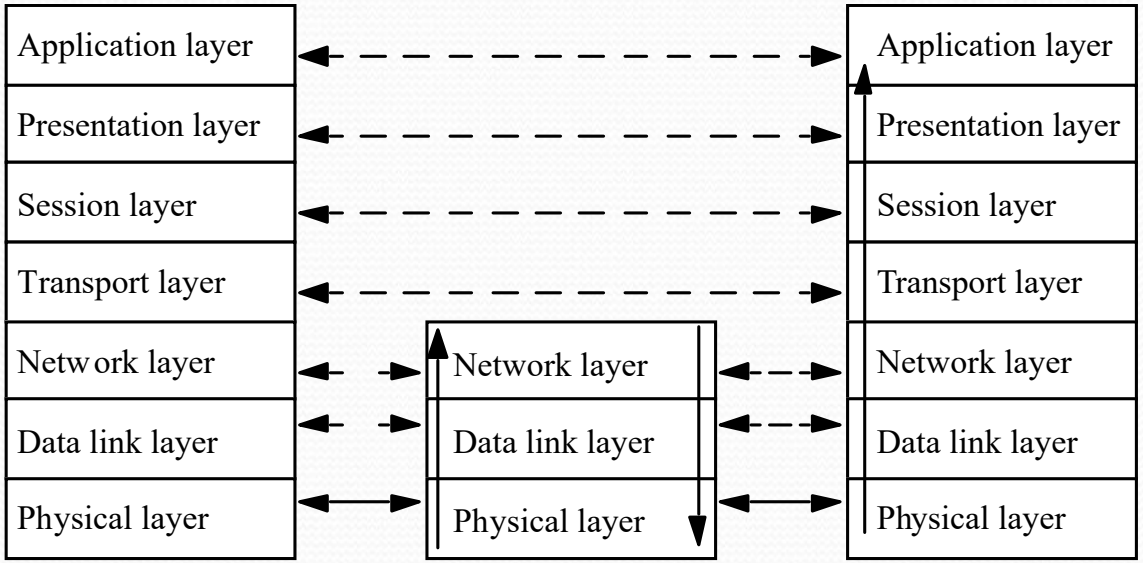


Multiple Encryption

- Double encipherment: $c = E_k(E_k(m))$
  - Effective key length is $2n$, if k, k' are length n
  - Problem: breaking it requires 2^{n+1} encryptions, not 2^{2n} encryptions
- Triple encipherment:
  - EDE mode: $c = E_k(D_k(E_k(m))$
    - Problem: chosen plaintext attack takes $O(2^n)$ time using 2^n ciphertexts
  - Triple encryption mode: $c = E_k(E_k(E_{k'}(m))$
    - Best attack requires $O(2^{2n})$ time, $O(2^n)$ memory

Networks and Cryptography

- ISO/OSI model
- Conceptually, each host has peer at each layer
  - Peers communicate with peers at same layer



Acknowledgement: Matt Bishop

Link and End-to-End Protocols

Link Protocol

End-to-End (or E2E) Protocol

Encryption

- Link encryption
  - Each host enciphers message so host at "next hop" can read it
  - Message can be read at intermediate hosts
- End-to-end encryption
  - Host enciphers message so host at other end of communication can read it
  - Message cannot be read at intermediate hosts

Cryptographic Considerations

- Link encryption
  - Each host shares key with neighbor
  - Can be set on per-host or per-host-pair basis
    - Windsor, stripe, seaview each have own keys
    - One key for (windsor, stripe); one for (stripe, seaview); one for (windsor, seaview)
- End-to-end
  - Each host shares key with destination
  - Can be set on per-host or per-host-pair basis
  - Message cannot be read at intermediate nodes

Traffic Analysis

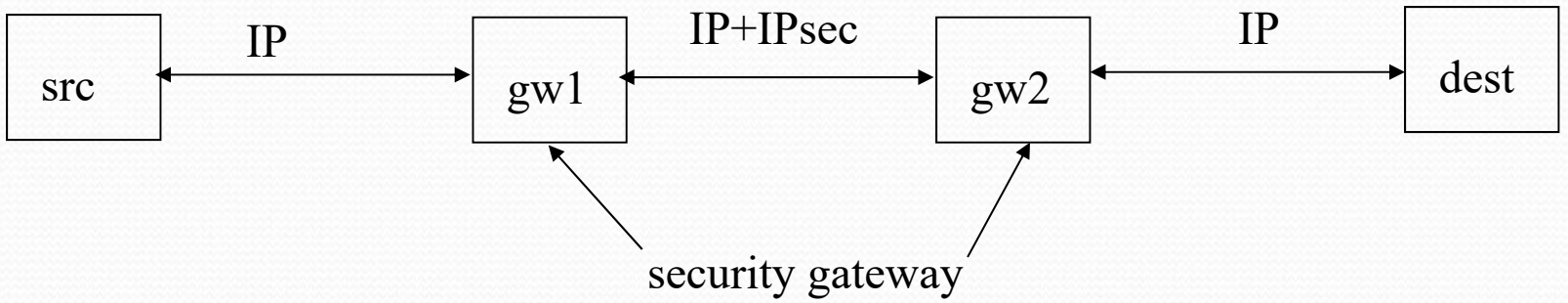
- Link encryption
  - Can protect headers of packets
  - Possible to hide source and destination
- End-to-end encryption
  - Cannot hide packet headers
    - Intermediate nodes need to route packet
  - Attacker can read source, destination

Example Protocols

- Privacy-Enhanced Electronic Mail (PEM)
  - Applications layer protocol
- IP Security (IPSec)
  - Network layer protocol

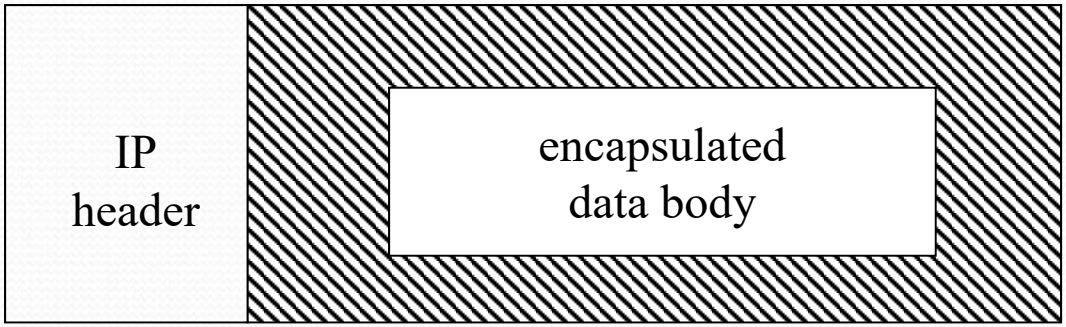
IPsec

- Network layer security
  - Provides confidentiality, integrity, authentication of endpoints, replay detection
- Protects all messages sent along a path



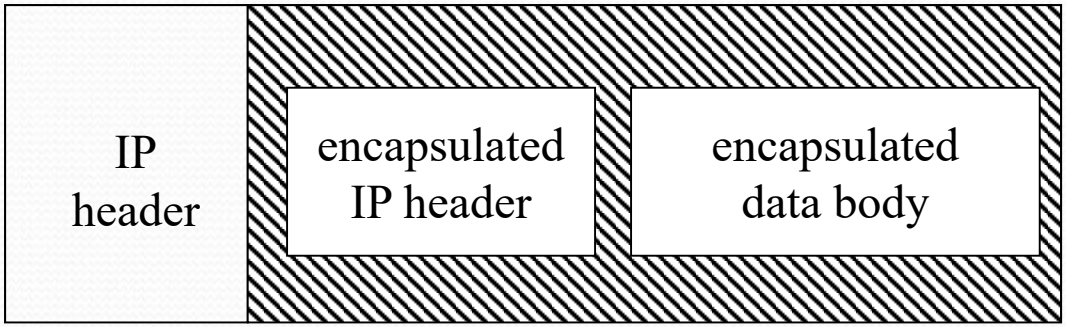
Acknowledgement: Matt Bishop

IPsec Transport Mode



- Encapsulate IP packet data area
- Use IP to send IPsec-wrapped data packet
- Note: IP header not protected

IPsec Tunnel Mode



- Encapsulate IP packet (IP header *and* IP data)
- Use IP to send IPsec-wrapped packet
- Note: IP header protected

IPsec Protocols

- Authentication Header (AH)
  - Message integrity
  - Origin authentication
  - Anti-replay
- Encapsulating Security Payload (ESP)
  - Confidentiality
  - Others provided by AH

Acknowledgement: Matt Bishop

IPsec Architecture

- Security Policy Database (SPD)
  - Says how to handle messages (discard them, apply security services, forward message unchanged)
  - SPD associated with network interface
  - SPD determines appropriate entry from packet attributes (search key)
    - Including source port and address, destination port and address, transport protocol

Example

- Goals
  - Discard SMTP packets from host 192.168.2.9
  - Forward packets from 192.168.19.7 without change
- SPD entries

```
src 192.168.2.9, dest 10.1.2.3 to 10.1.2.103, port 25, discard
src 192.168.19.7, dest 10.1.2.3 to 10.1.2.103, port 25, bypass
dest 10.1.2.3 to 10.1.2.103, port 25, apply IPsec
```

- Note: entries scanned in order
  - If no match for packet, it is discarded

Security Association (SA)

- Association between peers for security services
- Unidirectional
  - Can apply different services in either direction
- SA uses either ESP (Encapsulating Security Payload)or AH (Authentication Header); if both required, 2 SAs needed

Security Association

- Each SA is uniquely identified by three parameters:
  - *Security Parameters Index* (SPI)
  - *IP destination address*
  - *Security protocol identifier (AH or ESP)*

Security Association

- Security Parameters Index (SPI)
  - The SPI is a bit string assigned to the SA that has local significance only.
  - The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

Security Association

- Security Protocol Identifier
  - Indicates which IPSec protocol is in use on the SA
    - AH (Authentication only)
    - ESP (complete encryption and possibly Authentication)

SA Database (SAD)

- Entry describes SA; some fields for all packets:
  - AH algorithm identifier, keys
    - When SA uses AH
  - ESP encipherment algorithm identifier, keys
    - When SA uses confidentiality from ESP
  - ESP authentication algorithm identifier, keys
    - When SA uses authentication, integrity from ESP
  - SA lifetime (time for deletion or max byte count)
  - IPsec mode (tunnel, transport, either)

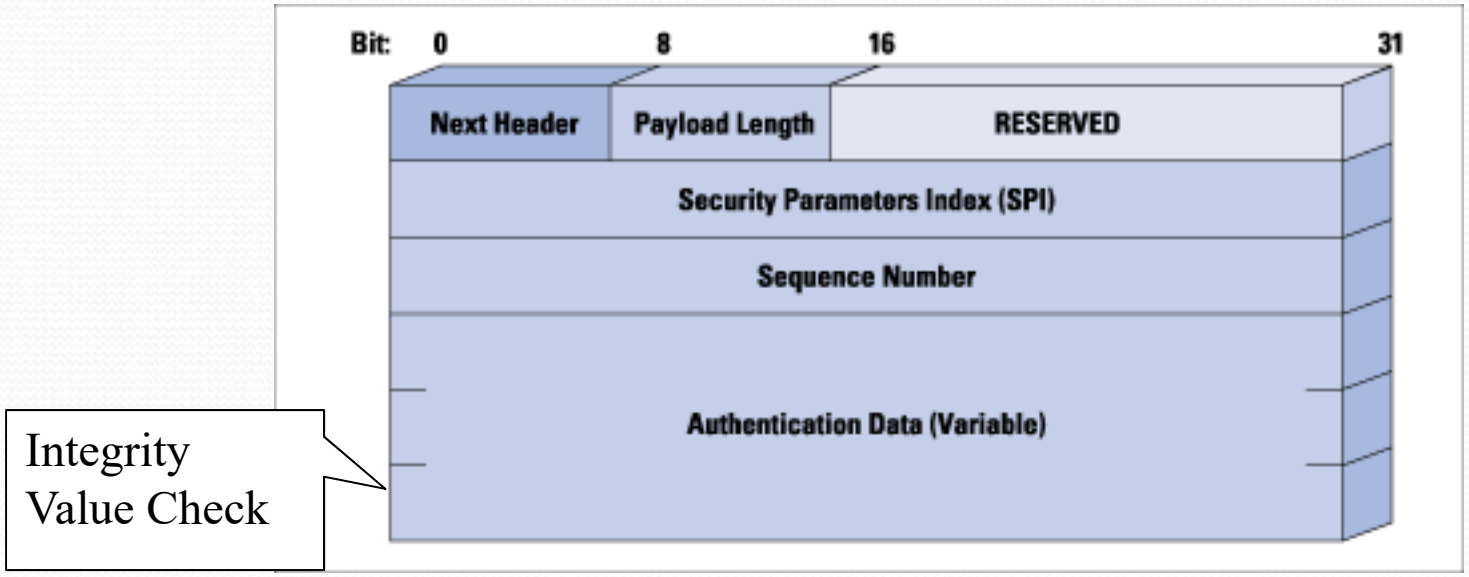
SAD Fields

- Antireplay (inbound only)
  - When SA uses antireplay feature
- Sequence number counter (outbound only)
  - Generates AH or ESP sequence number
- Sequence counter overflow field
  - Stops traffic over this SA if sequence counter overflows
- Aging variables
  - Used to detect time-outs

IPsec Architecture

- Packet arrives
- Look in SPD (Security Policy DB)
  - Find appropriate entry based on source port and address, destination port and address, and transport protocol (TCP / SMTP / …)
  - Get dest address, security protocol, SPI
- Find associated SA in SAD
  - Use dest address, security protocol (AH or ESP), SPI
  - Apply security services in SA (if any)

Authentication Header Protocol



- Two steps in handling
  - Check that replay is not occurring
  - Check authentication data

Sender

- Check sequence number will not cycle
- Increment sequence number
- Compute IVC (Integrity value check) of packet
  - Includes IP header, AH header, packet data
    - IP header: include all fields that will not change in transit; assume all others are o
    - AH header: authentication data field set to o for this
    - Packet data includes encapsulated data, higher level protocol data

Recipient

- Assume AH header found
- Get SPI, destination address
- Find associated SA in SAD
  - If no associated SA, discard packet
- If antireplay not used
  - Verify IVC is correct
    - If not, discard

Key Points

- Key management critical to effective use of cryptosystems
  - Different levels of keys (session *vs*. interchange)
- Block and stream cyphers
- One examples
  - IPsec