Chapter 10: Cipher Techniques

- Some Problems
- Types of Ciphers
- Networks
- Examples
Overview

• Problems
  – What can go wrong if you naively use ciphers
• Cipher types
  – Stream or block ciphers?
• Networks
  – Link vs end-to-end use
• Examples
  – Privacy-Enhanced Electronic Mail (PEM)
  – 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 $m_1 = \{ \text{BUY} \} e_{Bob}$, $m_2 = \{ \text{SELL} \} e_{Bob}$
- Cathy sees Alice send Bob $m_2$
- Cathy knows Alice sent SELL
May Not Be Obvious

• Digitized sound
  – Seems like far too many possible plaintexts
    • Initial calculations suggest $2^{32}$ such plaintexts
  – Analysis of redundancy in human speech reduced this to about 100,000 ($\approx 2^{17}$)
    • This is small enough to worry about precomputation attacks
Misordered 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 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)
  – More details later
What These Mean

• Use of strong cryptosystems, well-chosen (or random) keys not enough to be secure

• Other factors:
  – Protocols directing use of cryptosystems
  – Ancillary information added by protocols
  – Implementation (not discussed here)
  – Maintenance and operation (not discussed here)
Stream, Block Ciphers

- $E$ encipherment function
  - $E_k(b)$ encipherment of message $b$ with key $k$
  - In what follows, $m = b_1 b_2 \ldots$, each $b_i$ of fixed length
- Block cipher
  - $E_k(m) = E_k(b_1) E_k(b_2) \ldots$
- Stream cipher
  - $k = k_1 k_2 \ldots$
  - $E_k(m) = E_{k1}(b_1) E_{k2}(b_2) \ldots$
  - If $k_1 k_2 \ldots$ repeats itself, cipher is periodic and the length of its period is one cycle of $k_1 k_2 \ldots$
Examples

• Vigenère cipher
  – $b_i = 1$ character, $k = k_1k_2 \ldots$ where $k_i = 1$ character
  – Each $b_i$ enciphered using $k_i \mod \text{length}(k)$
  – Stream cipher

• DES
  – $b_i = 64$ bits, $k = 56$ bits
  – Each $b_i$ enciphered separately using $k$
  – Block cipher
Stream Ciphers

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

\[
m = 00101 \\
k = 10010 \\
c = 10111
\]

• 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 $r_0$
Operation

\[ r_0 \rightarrow \ldots \rightarrow r_{n-1} \rightarrow b_i \rightarrow c_i \]

\[ r_0' \rightarrow \ldots \rightarrow r_{n-1}' \]

\[ r_0 t_0 + \ldots + r_{n-1} t_{n-1} \]

\[ r_i' = r_{i-1}, \quad 0 < i \leq n \]
Example

- 4-stage LFSR; $t = 1001$

<table>
<thead>
<tr>
<th>$r$</th>
<th>$k_i$</th>
<th>New bit computation</th>
<th>New $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>0</td>
<td>$01\oplus00\oplus10\oplus01 = 0$</td>
<td>0001</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>$01\oplus00\oplus00\oplus11 = 1$</td>
<td>1000</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td>$11\oplus00\oplus00\oplus01 = 1$</td>
<td>1100</td>
</tr>
<tr>
<td>1100</td>
<td>0</td>
<td>$11\oplus10\oplus00\oplus01 = 1$</td>
<td>1110</td>
</tr>
<tr>
<td>1110</td>
<td>0</td>
<td>$11\oplus10\oplus10\oplus01 = 1$</td>
<td>1111</td>
</tr>
<tr>
<td>1111</td>
<td>1</td>
<td>$11\oplus10\oplus10\oplus11 = 0$</td>
<td>0111</td>
</tr>
<tr>
<td>1110</td>
<td>0</td>
<td>$11\oplus10\oplus10\oplus11 = 1$</td>
<td>1011</td>
</tr>
</tbody>
</table>

- Key sequence has period of 15 (010001111010110)
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; \( f(r_0, r_1, r_2, r_3) = (r_0 \& r_2) | r_3 \)

<table>
<thead>
<tr>
<th>( r )</th>
<th>( k_i )</th>
<th>new bit computation</th>
<th>new ( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>0</td>
<td>((1 &amp; 0))</td>
<td>0 = 0</td>
</tr>
<tr>
<td>0110</td>
<td>0</td>
<td>((0 &amp; 1))</td>
<td>0 = 0</td>
</tr>
<tr>
<td>0011</td>
<td>1</td>
<td>((0 &amp; 1))</td>
<td>1 = 1</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
<td>((1 &amp; 0))</td>
<td>1 = 1</td>
</tr>
<tr>
<td>1100</td>
<td>0</td>
<td>((1 &amp; 0))</td>
<td>0 = 0</td>
</tr>
<tr>
<td>0110</td>
<td>0</td>
<td>((0 &amp; 1))</td>
<td>0 = 0</td>
</tr>
<tr>
<td>0011</td>
<td>1</td>
<td>((0 &amp; 1))</td>
<td>1 = 1</td>
</tr>
</tbody>
</table>

– 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
Self-Synchronous Stream Cipher

• Take key from message itself (autokey)
• Example: Vigenère, key drawn from plaintext
  – keyXTHEBOYHASTHEBA
  – plaintextTHEBOYHASTHEBAG
  – ciphertextQALFPNFHSLALFCT

• Problem:
  – Statistical regularities in plaintext show in key
  – Once you get any part of the message, you can decipher more
Another Example

• Take key from ciphertext (*autokey*)
• Example: Vigenère, key drawn from ciphertext
  – *key* XQXBCQOVOVNGNRTT
  – *plaintext* THEBOYHASTHEBAG
  – *ciphertext* QXBCQOVVNGNRTTTM

• Problem:
  – Attacker gets key along with ciphertext, so deciphering is trivial
Variant

- Cipher feedback mode: 1 bit of ciphertext fed into $n$ bit register
  - Self-healing property: if ciphertext bit received incorrectly, it and next $n$ bits decipher incorrectly; but after that, the ciphertext bits decipher correctly
  - Need to know $k$, $E$ to decipher ciphertext
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 GHQMRSLIB 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
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

- ISØOSI model
- Conceptually, each host has peer at each layer
  - Peers communicate with peers at same layer
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
Examples

- **TELNET protocol**
  - Messages between client, server enciphered, and encipherment, decipherment occur only at these hosts
  - End-to-end protocol

- **PPP Encryption Control Protocol**
  - Host gets message, deciphers it
    - Figures out where to forward it
    - Enciphers it in appropriate key and forwards it
  - Link protocol
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
    • Note: may be able to deduce this from traffic flows

• 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
Goals of PEM

1. Confidentiality
   • Only sender and recipient(s) can read message
2. Origin authentication
   • Identify the sender precisely
3. Data integrity
   • Any changes in message are easy to detect
4. Non-repudiation of origin
   • Whenever possible …
Message Handling System

MTA → MTA → MTA

UA → MTA → UA

UA → MTA → UA

User Agents
Message Transfer Agents
Design Principles

• Do not change related existing protocols
  – Cannot alter SMTP

• Do not change existing software
  – Need compatibility with existing software

• Make use of PEM optional
  – Available if desired, but email still works without them
  – Some recipients may use it, others not

• Enable communication without prearrangement
  – Out-of-bands authentication, key exchange problematic
Basic Design: Keys

- Two keys
  - Interchange keys tied to sender, recipients and is static (for some set of messages)
    - Like a public/private key pair
    - Must be available before messages sent
  - Data exchange keys generated for each message
    - Like a session key, session being the message
Basic Design: Sending

Confidentiality

- $m$ message
- $k_s$ data exchange key
- $k_B$ Bob’s interchange key

\[ \{ m \} k_s \parallel \{ k_s \} k_B \]

Alice \hspace{1cm} Bob
Basic Design: Integrity

Integrity and authentication:

- $m$ message
- $h(m)$ hash of message $m$ —Message Integrity Check (MIC)
- $k_A$ Alice’s interchange key

```
m \{ h(m) \} k_A
```

Alice $\rightarrow$ Bob

Non-repudiation: if $k_A$ is Alice’s private key, this establishes that Alice’s private key was used to sign the message
Basic Design: Everything

Confidentiality, integrity, authentication:
• Notations as in previous slides
• If $k_A$ is private key, get non-repudiation too

\[
\{ m \} k_s \parallel \{ h(m) \} k_A \parallel \{ k_s \} k_B
\]

Alice --------> Bob
Practical Considerations

• Limits of SMTP
  – Only ASCII characters, limited length lines

• Use encoding procedure
  1. Map local char representation into canonical format
     – Format meets SMTP requirements
  2. Compute and encipher MIC over the canonical format; encipher message if needed
  3. Map each 6 bits of result into a character; insert newline after every 64th character
  4. Add delimiters around this ASCII message
Problem

- Recipient without PEM-compliant software cannot read it
  - If only integrity and authentication used, should be able to read it
- Mode MIC-CLEAR allows this
  - Skip step 3 in encoding procedure
  - Problem: some MTAs add blank lines, delete trailing white space, or change end of line character
  - Result: PEM-compliant software reports integrity failure
PEM vs. PGP

- Use different ciphers
  - PGP uses IDEA cipher
  - PEM uses DES in CBC mode
- Use different certificate models
  - PGP uses general “web of trust”
  - PEM uses hierarchical certification structure
- Handle end of line differently
  - PGP remaps end of line if message tagged “text”, but leaves them alone if message tagged “binary”
  - PEM always remaps end of line
IPsec

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

- Security Policy Database (SPD)
  - Says how to handle messages (discard them, add security services, forward message unchanged)
  - SPD associated with network interface
  - SPD determines appropriate entry from packet attributes
    - Including source, destination, 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
IPsec Architecture

- Security Association (SA)
  - Association between peers for security services
    - Identified uniquely by dest address, security protocol (AH or ESP), unique 32-bit number (security parameter index, or SPI)
  - Unidirectional
    - Can apply different services in either direction
  - SA uses either ESP or AH; if both required, 2 SAs needed
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
  – Find appropriate entry
  – Get dest address, security protocol, SPI
• Find associated SA in SAD
  – Use dest address, security protocol, SPI
  – Apply security services in SA (if any)
SA Bundles and Nesting

• Sequence of SAs that IPsec applies to packets
  – This is a SA bundle

• Nest tunnel mode SAs
  – This is iterated tunneling
Example: Nested Tunnels

• Group in A.org needs to communicate with group in B.org

• Gateways of A, B use IPsec mechanisms
  – But the information must be secret to everyone except the two groups, even secret from other people in A.org and B.org

• Inner tunnel: a SA between the hosts of the two groups

• Outer tunnel: the SA between the two gateways
Example: Systems

hostA.A.org

Internet

gwA.A.org

gwB.B.org

hostB.B.org

SA in tunnel mode (outer tunnel)

SA in tunnel mode (inner tunnel)
Example: Packets

- Packet generated on hostA
- Encapsulated by hostA’s IPsec mechanisms
- Again encapsulated by gwA’s IPsec mechanisms
  - Above diagram shows headers, but as you go left, everything to the right would be enciphered and authenticated, *etc.*
AH Protocol

• Parameters in AH header
  – Length of header
  – SPI of SA applying protocol
  – Sequence number (anti-replay)
  – Integrity value check

• Two steps
  – Check that replay is not occurring
  – Check authentication data
**Sender**

- Check sequence number will not cycle
- Increment sequence number
- Compute IVC of packet
  - Includes IP header, AH header, packet data
    - IP header: include all fields that will not change in transit; assume all others are 0
    - AH header: authentication data field set to 0 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
Recipient, Using Antireplay

- Check packet beyond low end of sliding window
- Check IVC of packet
- Check packet’s slot not occupied
  - If any of these is false, discard packet

```
    □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
AH Miscellany

• All implementations must support:
  HMAC_MD5
  HMAC_SHA-1
• May support other algorithms
ESP Protocol

- Parameters in ESP header
  - SPI of SA applying protocol
  - Sequence number (anti-replay)
  - Generic “payload data” field
  - Padding and length of padding
    - Contents depends on ESP services enabled; may be an initialization vector for a chaining cipher, for example
    - Used also to pad packet to length required by cipher
  - Optional authentication data field
Sender

- Add ESP header
  - Includes whatever padding needed
- Encipher result
  - Do not encipher SPI, sequence numbers
- If authentication desired, compute as for AH protocol except over ESP header, payload and not encapsulating IP header
Recipient

- Assume ESP header found
- Get SPI, destination address
- Find associated SA in SAD
  - If no associated SA, discard packet
- If authentication used
  - Do IVC, antireplay verification as for AH
    - Only ESP, payload are considered; *not* IP header
    - Note authentication data inserted after encipherment, so no deciphering need be done
Recipient

- If confidentiality used
  - Decipher enciphered portion of ESP header
  - Process padding
  - Decipher payload
  - If SA is transport mode, IP header and payload treated as original IP packet
  - If SA is tunnel mode, payload is an encapsulated IP packet and so is treated as original IP packet
ESP Miscellany

• Must use at least one of confidentiality, authentication services
• Synchronization material must be in payload
  – Packets may not arrive in order, so if not, packets following a missing packet may not be decipherable
• Implementations of ESP assume classical cryptosystem
  – Implementations of public key systems usually far slower than implementations of classical systems
  – Not required
More ESP Miscellany

- All implementations must support (encipherment algorithms):
  - DES in CBC mode
  - NULL algorithm (identity; no encipherment)
- All implementations must support (integrity algorithms):
  - HMAC_MD5
  - HMAC_SHA-1
  - NULL algorithm (no MAC computed)
- Both cannot be NULL at the same time
Which to Use: PEM, IPsec

• What do the security services apply to?
  – If applicable to one application and application layer mechanisms available, use that
    • PEM for electronic mail
  – If more generic services needed, look to lower layers
    • IPsec for network layer, either end-to-end or link mechanisms, for connectionless channels as well as connections
  – If endpoint is host, IPsec sufficient; if endpoint is user, application layer mechanism such as PEM needed
Key Points

• Key management critical to effective use of cryptosystems
  – Different levels of keys (session vs. interchange)
• Keys need infrastructure to identify holders, allow revoking
  – Key escrow complicates infrastructure
• Digital signatures provide integrity of origin and content
  Much easier with public key cryptosystems than with classical cryptosystems