Chapter 7 – Confidentiality Using Symmetric Encryption

Confidentiality using Symmetric Encryption

- Assume that traditional symmetric encryption is used to provide message confidentiality
- consider typical scenario
- What are the possible points of vulnerability

Typical Scenario

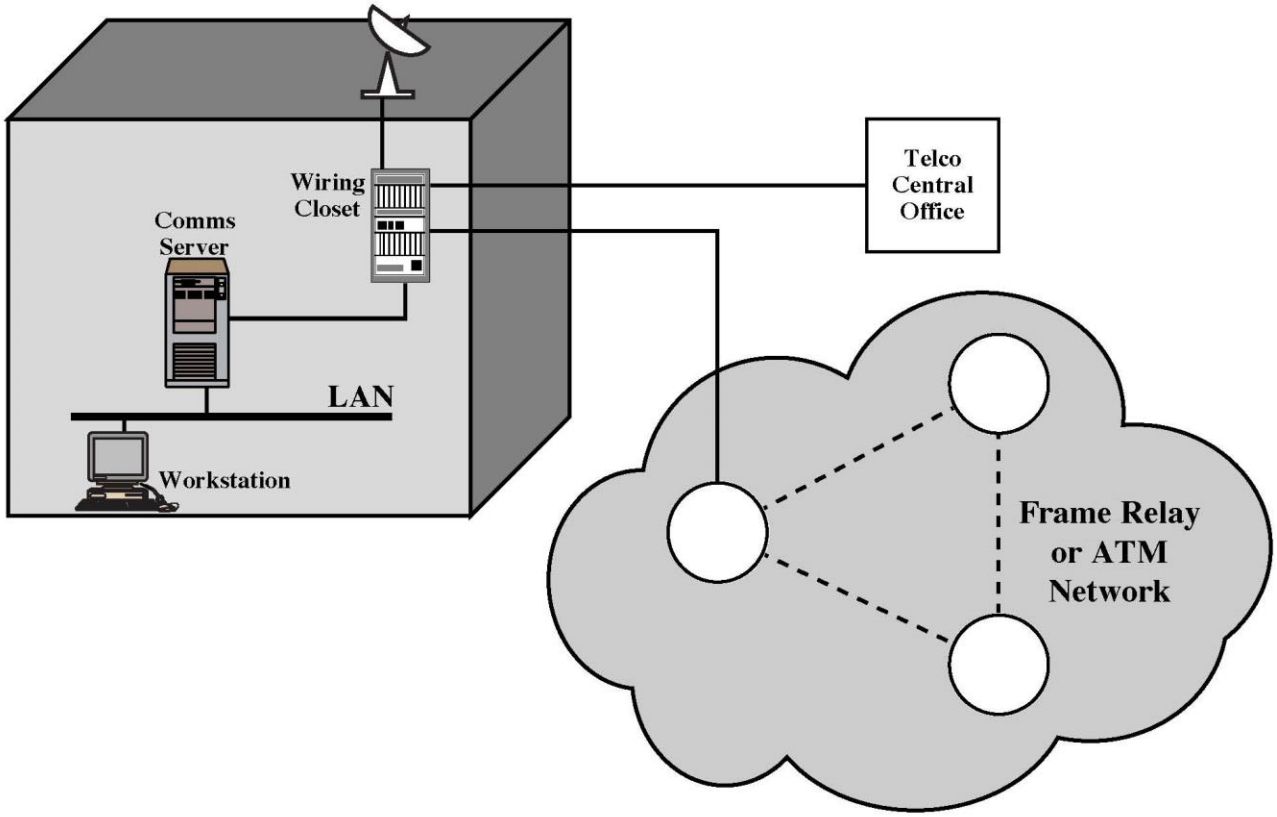


Figure 7.1 Points of Vulnerability

Points of attacks

- consider attacks and placement in this scenario
  - snooping from another workstation
    - LAN is a broadcast network
    - Traffic visible to all workstations in the LAN
  - use dial-in to LAN or server to snoop
    - If a server or a workstation offers dial-in service
  - router can be vulnerable
    - If one has physical access to the router
  - monitor and/or modify traffic one external links

Placement of Security Devices

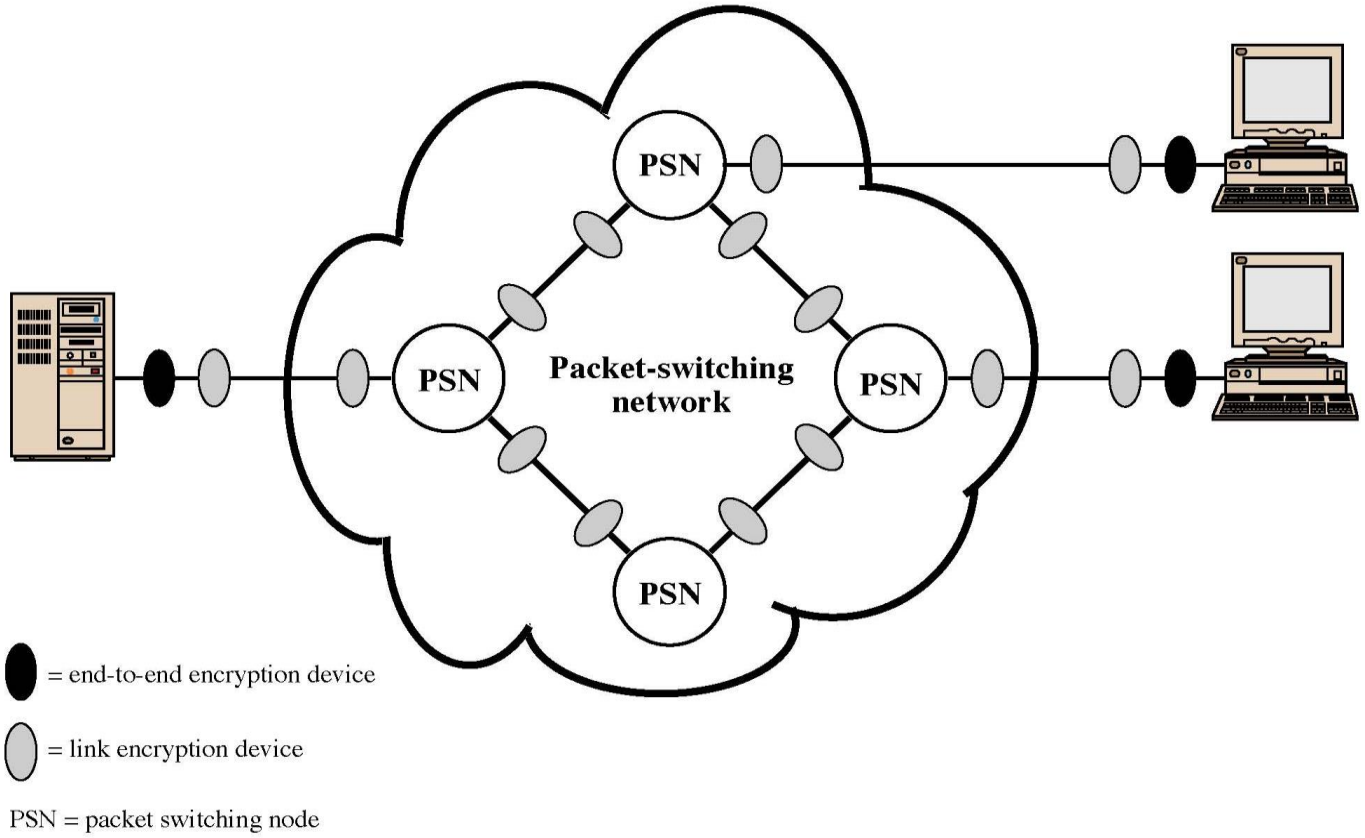


Figure 7.2 Encryption Across a Packet-Switching Network

Two major placement alternatives

- **link encryption**
  - encryption occurs independently on every link
  - implies must decrypt traffic between links
  - One key per (node, node) pair
  - Message exposed in nodes
  - Transparent to user, done in hardware
- **end-to-end encryption**
  - encryption occurs between original source and final destination
  - One key per user pair
  - Message encrypted in nodes
  - User selects hardware, software implementation

Traffic Analysis

- when using end-to-end encryption must leave headers in clear
  - so network can correctly route information
- hence although contents protected, traffic pattern flows are not

Key Distribution

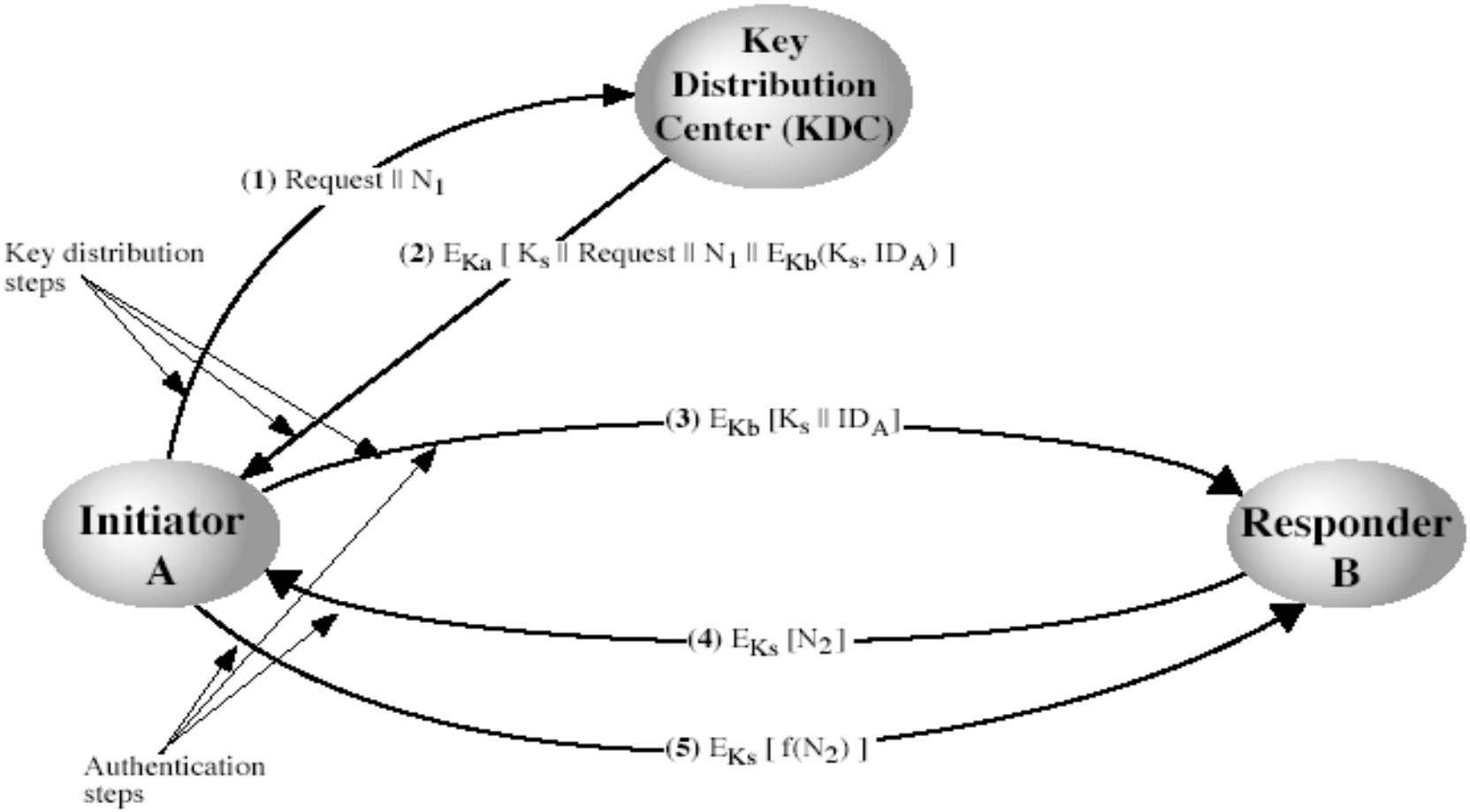
- symmetric schemes require both parties to share a common secret key
- issue is how to securely distribute this key
- often secure system failure due to a break in the key distribution scheme

Key Distribution

- given parties A and B have various **key distribution** alternatives:
  1. A can select key and physically deliver to B
  2. third party can select & deliver key to A & B
  3. if A & B have communicated previously can use previous key to encrypt a new key
  4. if A & B have secure communications with a third party C, C can relay key between A & B

As number of parties grow, some variant of 4 is only practical solution.

Key Distribution Scenario



Random Numbers

- many uses of **random numbers** in cryptography
  - Ns in authentication protocols to prevent replay
  - session keys
  - public key generation
  - keystream for a one-time pad
- in all cases its critical that these values be
  - statistically random
    - with uniform distribution, independent
  - unpredictable cannot infer future sequence on previous values

Natural Random Noise

- best source is natural randomness in real world
- find a random event and monitor
- generally need special h/w to do this
  - eg. radiation counters, radio noise, audio noise, thermal noise, leaky capacitors, mercury discharge tubes etc

Published Sources

- a few published collections of random numbers
- Rand Co, in 1955, published 1 million numbers
  - generated using an electronic roulette wheel
  - has been used in some cipher designs cf Khafre
- earlier Tippett in 1927 published a collection
- issues are that:
  - these are limited
  - too well-known for most uses

Pseudorandom Number Generators (PRNGs)

- algorithmic technique to create “random numbers”
  - although not truly random

Linear Congruential Generator

- common iterative technique using:

$$X_{n+1} = (aX_n + c) \bmod m$$

- given suitable values of parameters can produce a long random-like sequence
- note that an attacker can reconstruct sequence given a small number of values

Using Block Ciphers as Stream Ciphers

- can use block cipher to generate numbers
- use Counter Mode

  $X_i = \mathrm{E}_{Km}[\mathrm{i}]$

- use Output Feedback Mode

  $X_i = \mathrm{E}_{Km}[X_{i-1}]$

- ANSI standard, uses output feedback 3-DES

Blum Blum Shub Generator

- use least significant bit from iterative equation:
  - Get prime p, q, such that p,q=3 mod 4
  - Get n=p.q, and a random number s, gcd(s,n)=1
  - $X_0 = s^2 \bmod n$
  - $x_{i+1} = x_i^2 \bmod n$
- Output: binary sequence: 110011100001 (table 7.2)
- is unpredictable given any run of bits
- Passes the *next-bit test*
  - No poly-time algorithm that can predict the next bit with p>1/2
- slow, since very large numbers must be used
- too slow for cipher use, good for key generation

Summary

- have considered:
  - use of symmetric encryption to protect confidentiality
  - need for good key distribution
  - use of trusted third party KDC
  - random number generation