Chapter 7

Confidentiality Using Symmetric Encryption

Confidentiality using Symmetric Encryption

- traditionally symmetric encryption is used to provide message confidentiality
- consider typical scenario
  - workstations on LANs access other workstations & servers on LAN
  - LANs interconnected using switches/routers
  - with external lines or radio/satellite links
- consider attacks and placement in this scenario
  - snooping from another workstation
  - use dial-in to LAN or server to snoop
  - use external router link to enter & snoop
  - monitor and/or modify traffic one external links

Confidentiality using Symmetric Encryption

- have two major placement alternatives
- **link encryption**
  - encryption occurs independently on every link
  - implies must decrypt traffic between links
  - requires many devices, but paired keys
- **end-to-end encryption**
  - encryption occurs between original source and final destination
  - need devices at each end with shared keys

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
- ideally want both at once
  - end-to-end protects data contents over entire path and provides authentication
  - link protects traffic flows from monitoring

Placement of Encryption

- can place encryption function at various layers in OSI Reference Model
  - link encryption occurs at layers 1 or 2
  - end-to-end can occur at layers 3, 4, 6, 7
  - as move higher less information is encrypted but it is more secure though more complex with more entities and keys

Traffic Analysis

- is monitoring of communications flows between parties
  - useful both in military & commercial spheres
  - can also be used to create a covert channel
- link encryption obscures header details
  - but overall traffic volumes in networks and at end-points is still visible
- traffic padding can further obscure flows
  - but at cost of continuous traffic

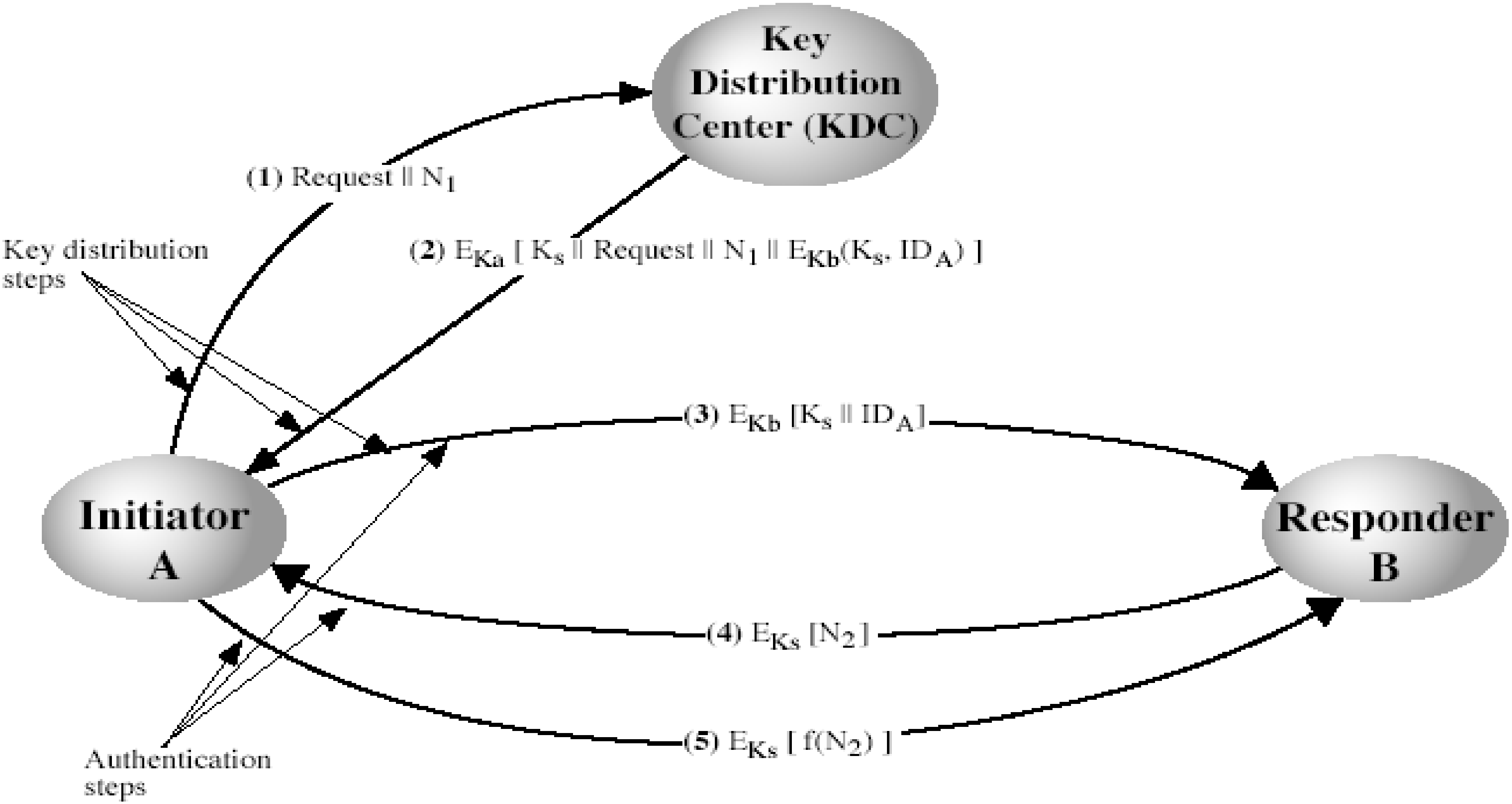
Key Distribution

- 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

Key Distribution Scenario



Key Distribution Issues

- hierarchies of KDC's required for large networks, but must trust each other
- session key lifetimes should be limited for greater security
- use of automatic key distribution on behalf of users, but must trust system
- use of decentralized key distribution
- controlling purposes keys are used for

Random Numbers

- many uses of **random numbers** in cryptography
  - nonces 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 regular but random event and monitor
- do generally need special h/w to do this
  - eg. radiation counters, radio noise, audio noise, thermal noise in diodes, leaky capacitors, mercury discharge tubes etc
- starting to see such h/w in new CPU's
- problems of **bias** or uneven distribution in signal
  - have to compensate for this when sample and use
  - best to only use a few noisiest bits from each sample

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
  - can pass many tests of “randomness”

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
- suitable criteria to have are:
  - function generates a full-period
  - generated sequence should appear random
  - efficient implementation with 32-bit arithmetic
- 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

  $X_i = E_{Km}[i]$
- use Output Feedback Mode

  $X_i = E_{Km}[X_{i-1}]$
- ANSI X9.17 PRNG
  - uses date-time + seed inputs and 3 triple-DES encryptions to generate new seed & random

Blum Blum Shub Generator

- based on public key algorithms
- use least significant bit from iterative equation:
  - $x_{i+1} = x_i^2 \bmod n$
  - where n=p.q, and primes p,q=3 mod 4
- unpredictable, passes **next-bit** test
- security rests on difficulty of factoring N
- is unpredictable given any run of bits
- 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's
  - random number generation