Plan

- Symmetric Key Distribution
- Random Number Generation

Key Management and Distribution

- Topics of cryptographic key management / key distribution are complex
  - cryptographic, protocol, & management issues
- Symmetric schemes require both parties to share a common secret key
- Public key schemes require parties to acquire valid public keys
- Have concerns with doing both

Symmetric key distribution

- Recall, we noted two important requirements for secure use of symmetric encryption:
  - a strong encryption algorithm
  - a secret key known only to sender / receiver
- Main purpose of key distribution to ensure that keys are private between the sender and receiver.
- How do you achieve key distribution?

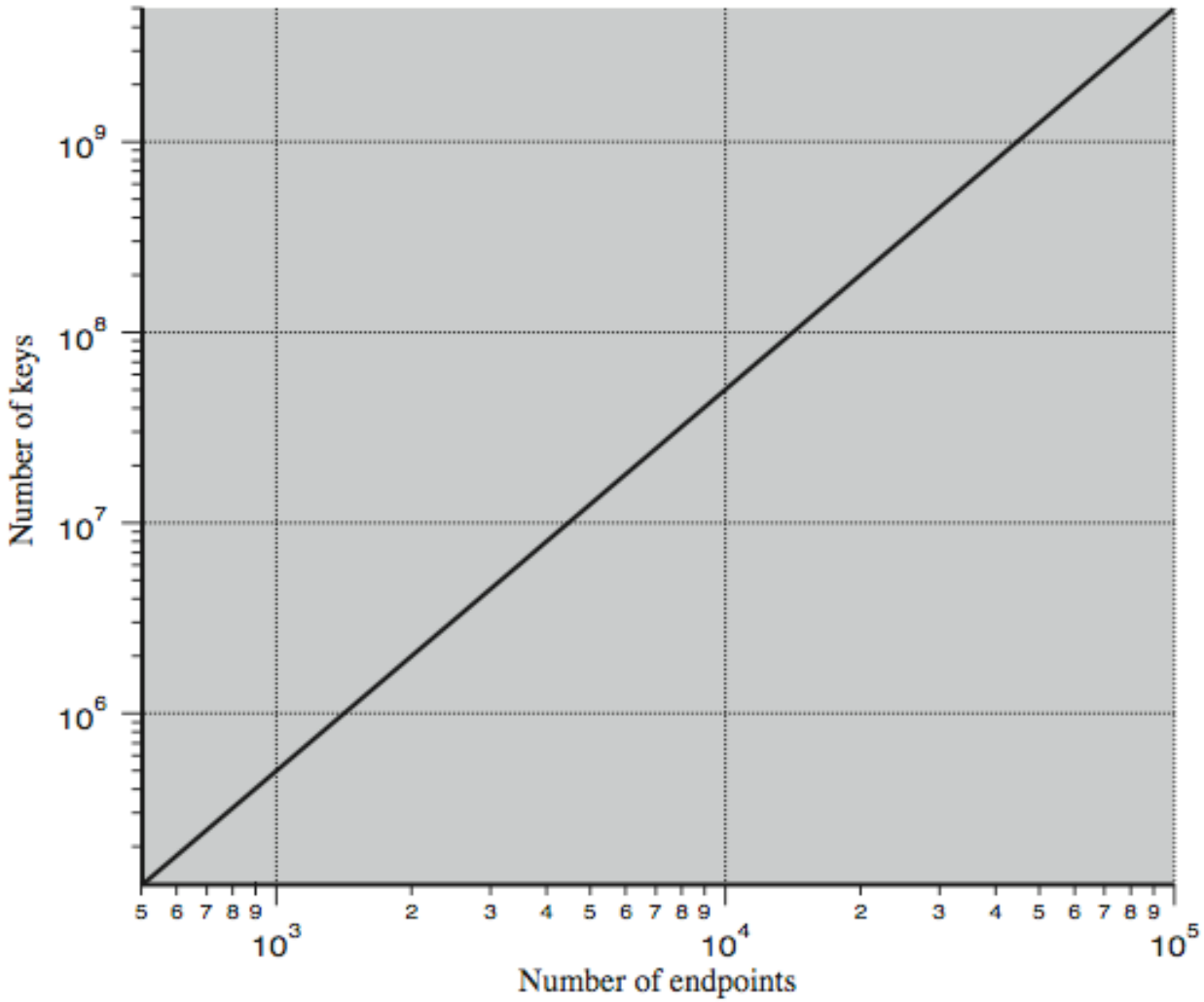
Key Distribution

- Symmetric Key schemes require both parties to share a common secret key and hence the main issue is how to securely distribute this key whilst protecting it from others
- Frequent key changes can be desirable
- 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 Task



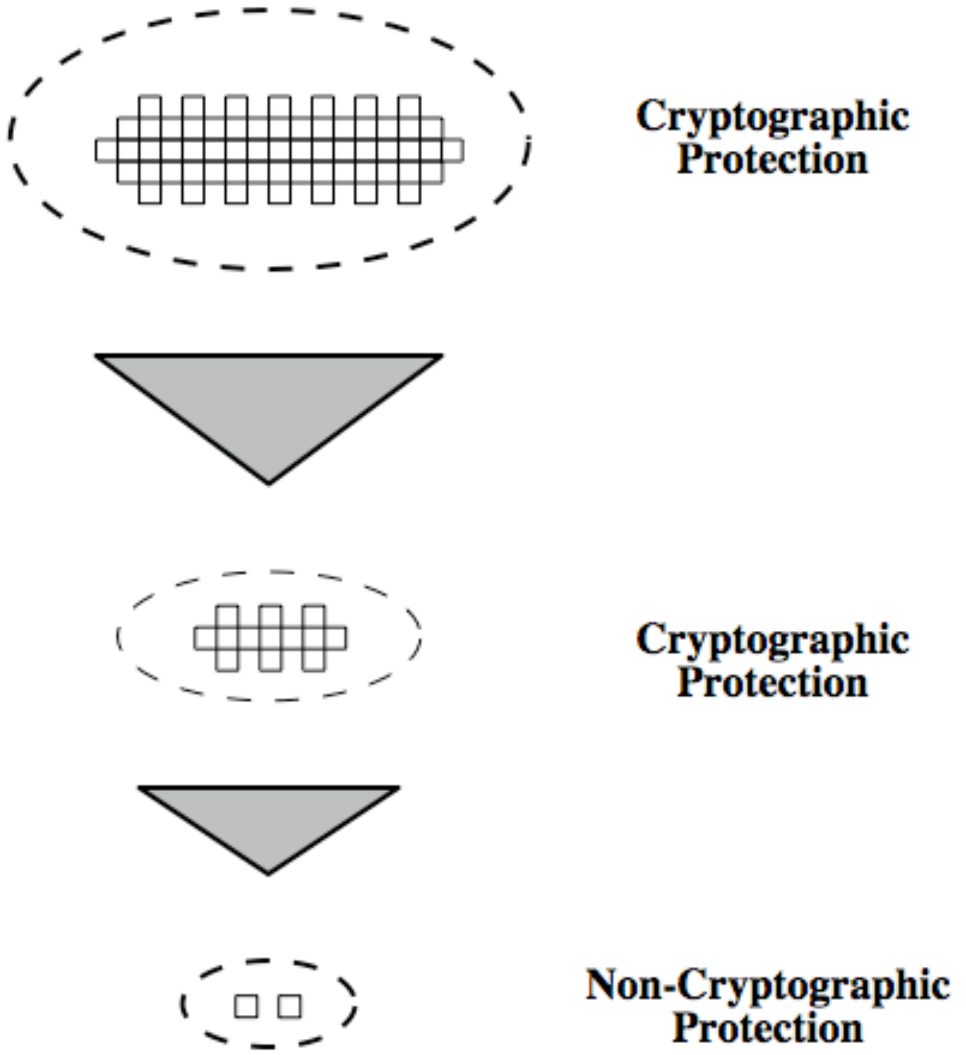
Key Hierarchy

Session key

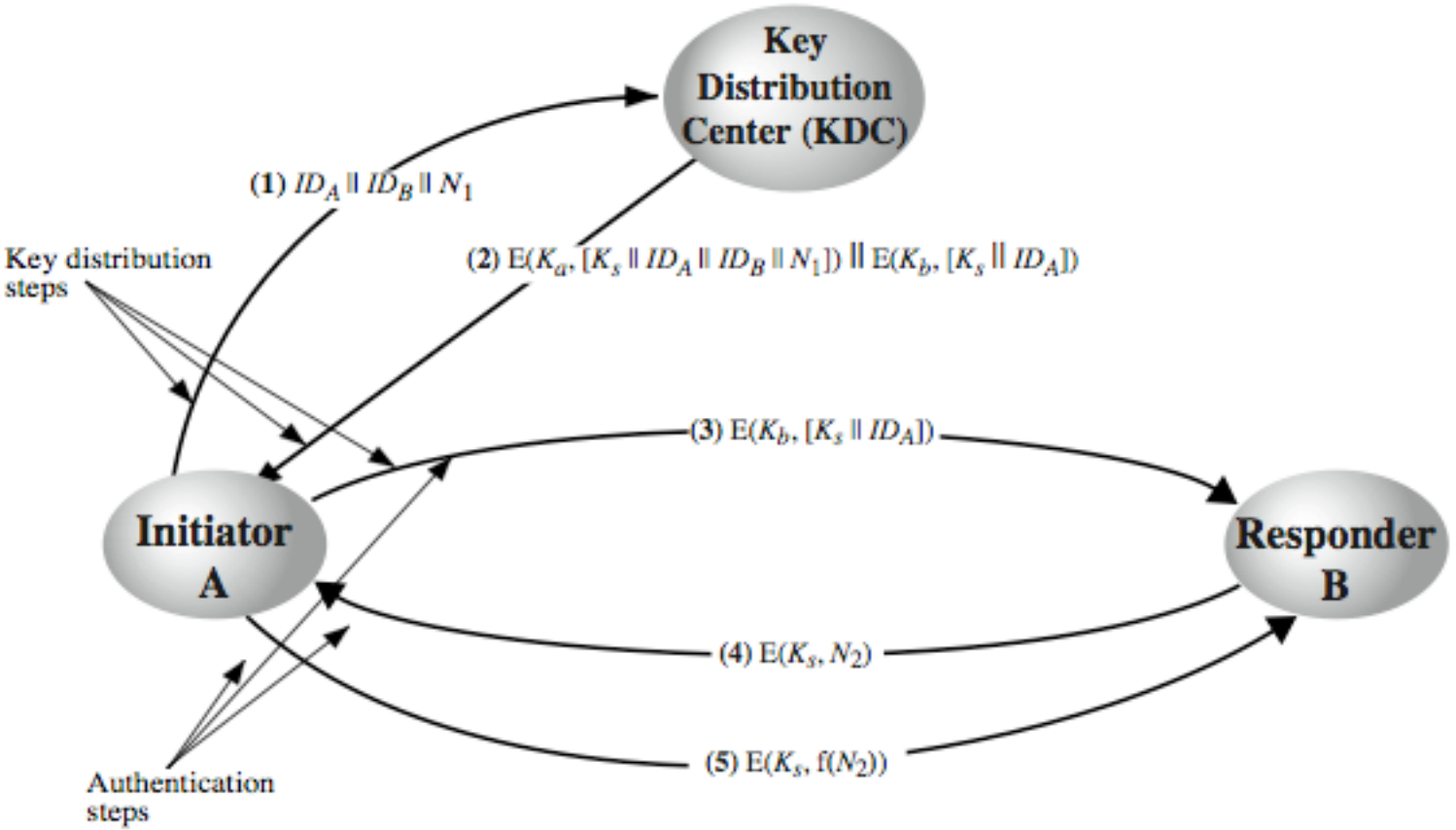
- temporary key
- used for encryption of data between users
- for one logical session then discarded

Master key

- used to encrypt session keys
- shared by user & key distribution center



Key Distribution Scenario



Key Distribution Issues

- For communication among entities within the same local domain, the local KDC is responsible for key distribution
  - If two entities in different domains desire a shared key, then the corresponding local KDC's can communicate through a global KDC
- The hierarchical concept can be extended to three or more layers
- Scheme minimizes the effort involved in master key distribution because most master keys are those shared by a local KDC with its local entities
  - Limits the range of a faulty or subverted KDC to its local area only

Session Key Lifetime

For connection-oriented protocols one choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session

A security manager must balance competing considerations:

- The more frequently session keys are exchanged, the more secure they are
- The distribution of session keys delays the start of any exchange and places a burden on network capacity

For a connectionless protocol there is no explicit connection initiation or termination, thus it is not obvious how often one needs to change the session key

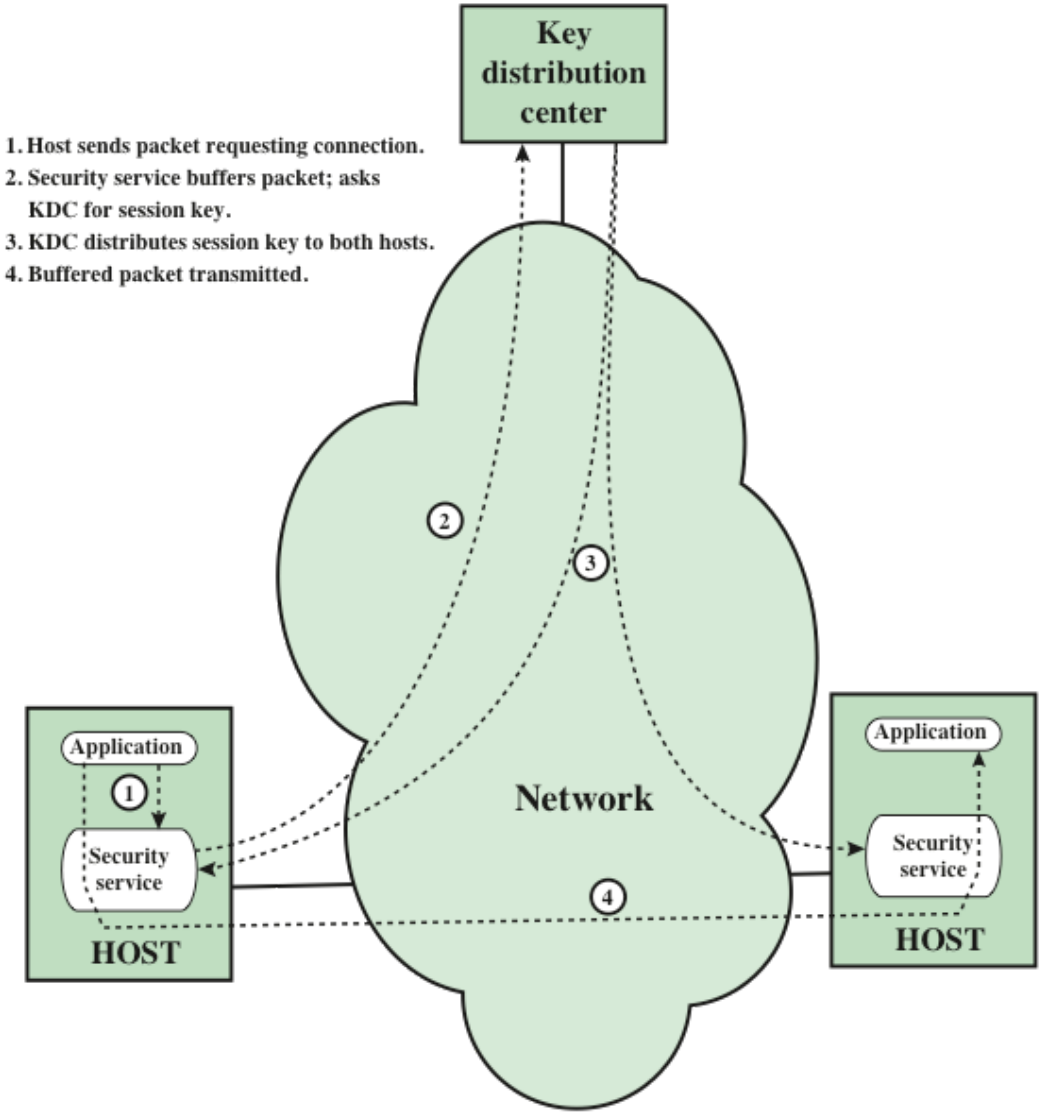


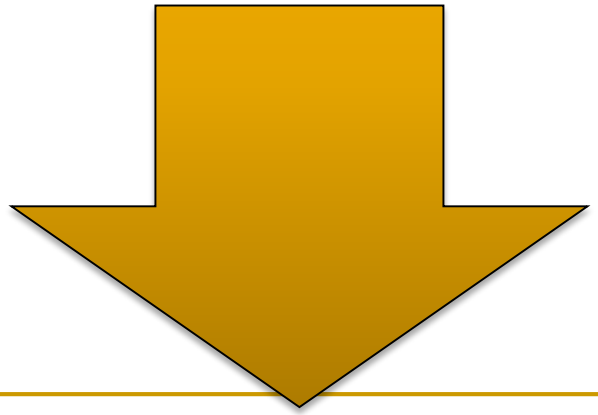
Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

Controlling Key Usage

- The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed
- It also may be desirable to impose some control on the way in which automatically distributed keys are used
  - For example, in addition to separating master keys from session keys, we may wish to define different types of session keys on the basis of use

Key Controls

- Associate a tag with each key
  - For use with DES and makes use of the extra 8 bits in each 64-bit DES key
  - The eight non-key bits ordinarily reserved for parity checking form the key tag
  - Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection



Drawbacks:

- The tag length is limited to 8 bits, limiting its flexibility and functionality
- Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled

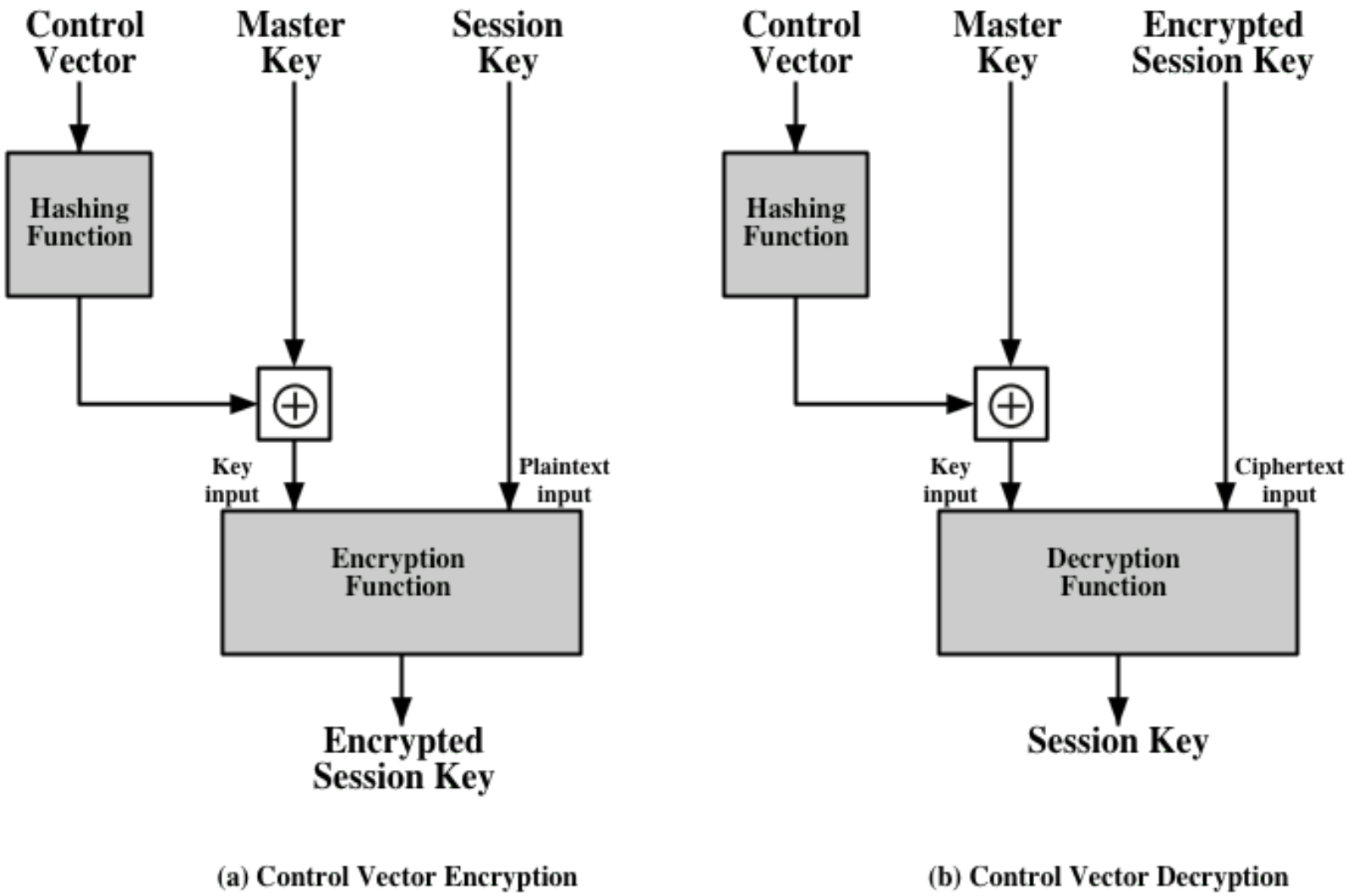
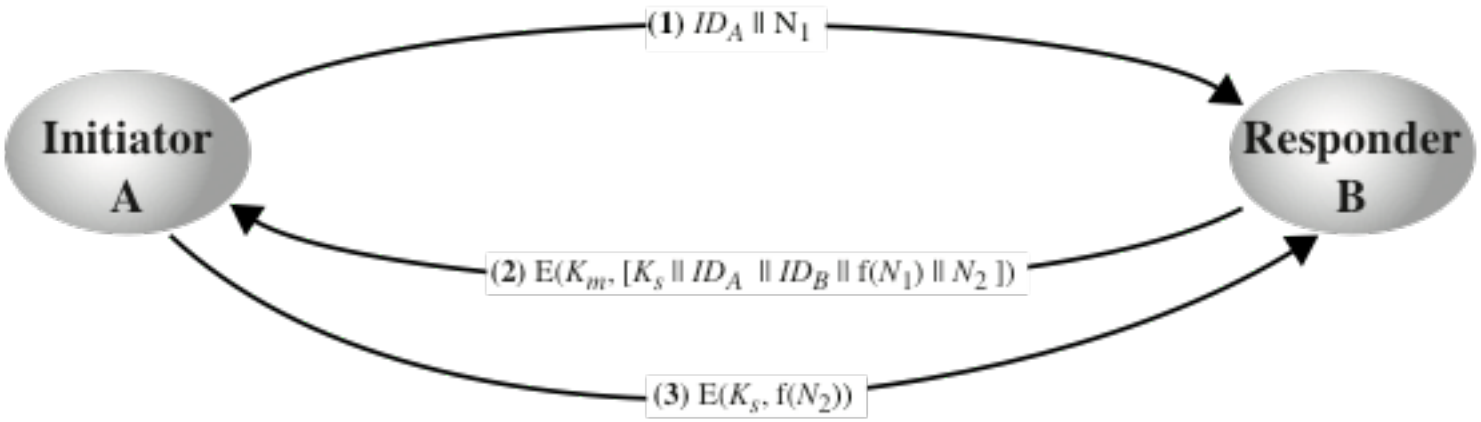


Figure 14.6 Control Vector Encryption and Decryption

A decentralized key agreement Scenario



Assumes that all end users are able to securely obtain keys to talk to each other;

Figure 14.5 Decentralized Key Distribution

Symmetric Key Distribution Using Public Keys

Can we use public key systems to obtain symmetric key secrets?

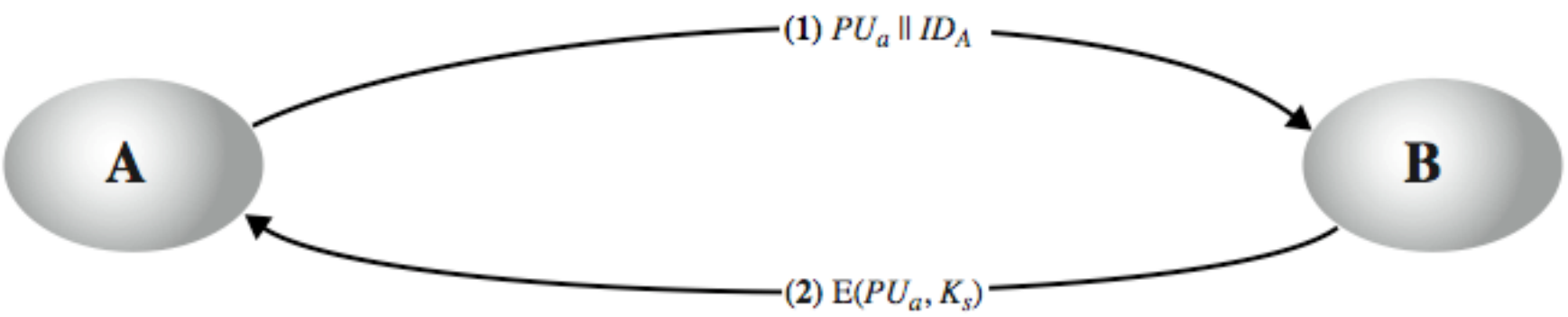
Issues:

Public key cryptosystems are inefficient

- so almost never use for direct data encryption
- rather use to encrypt secret keys for distribution (Hybrid Schemes)

Simple Secret Key Distribution

- Merkle proposed this very simple scheme
  - Allows secure communications
  - No keys before/after exist
- Does this solves the problem of Trust?



Man-in-the-Middle Attack

This very simple scheme is vulnerable to an active man-in-the-middle attack

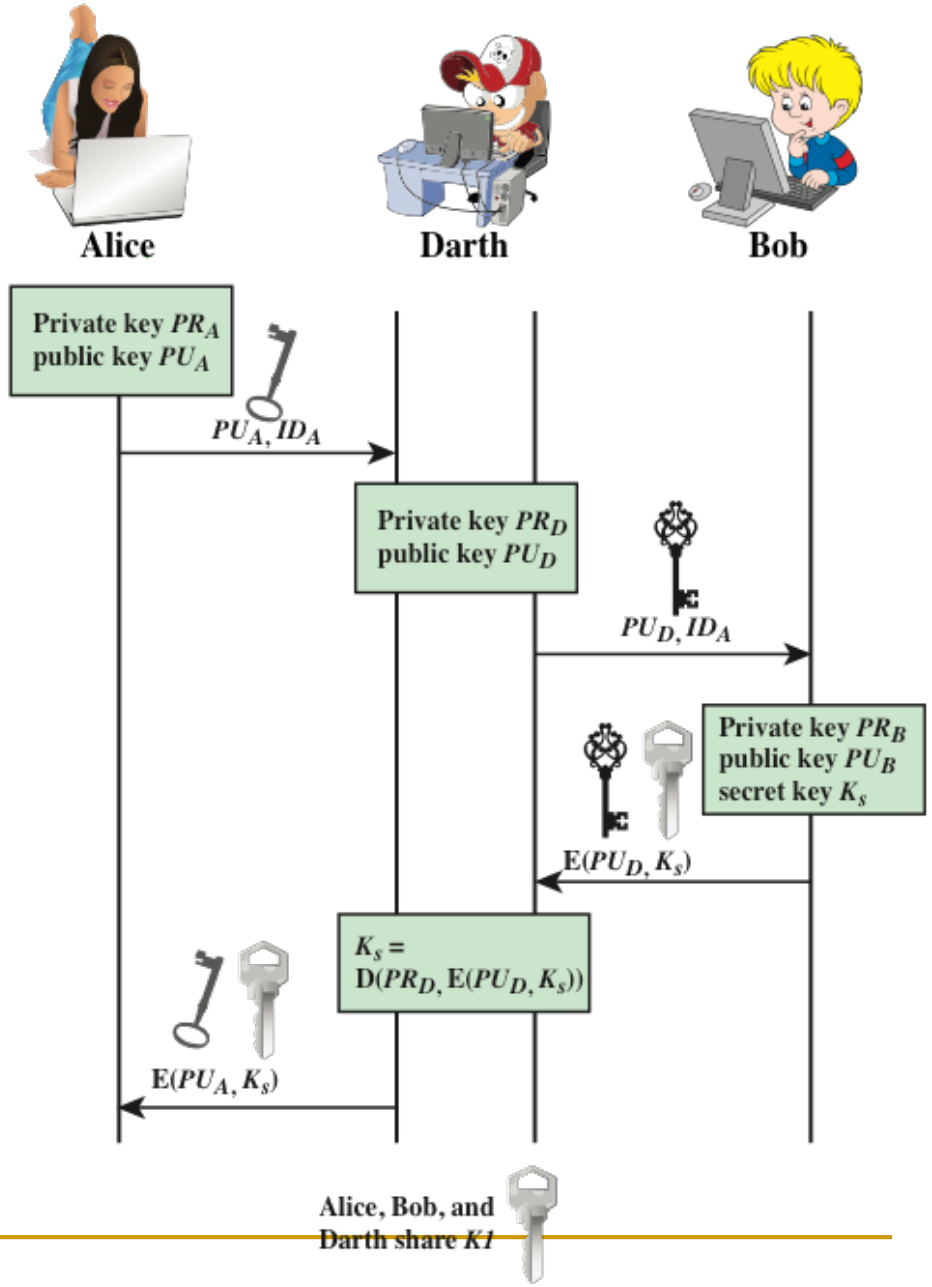
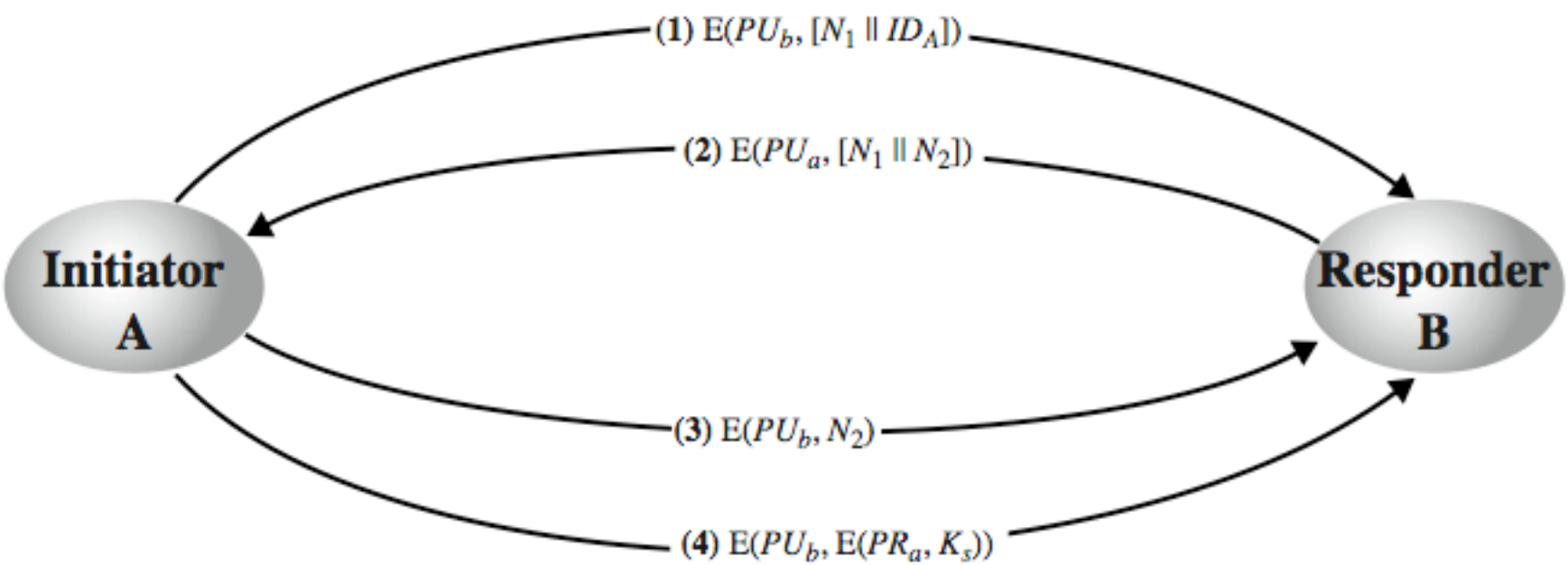


Figure 14.8 Another Man-in-the-Middle Attack

Secret Key Distribution with Confidentiality and Authentication



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, uniform distribution, independent
  - unpredictability of future values from previous values

Pseudorandom Number Generators (PRNGs)

- Often use deterministic algorithmic techniques to create “random numbers”
  - although are not truly random
  - can pass many tests of “randomness”
- Known as “pseudorandom numbers”
- Created by “Pseudorandom Number Generators (PRNGs)”

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
- have possibilities for making this harder

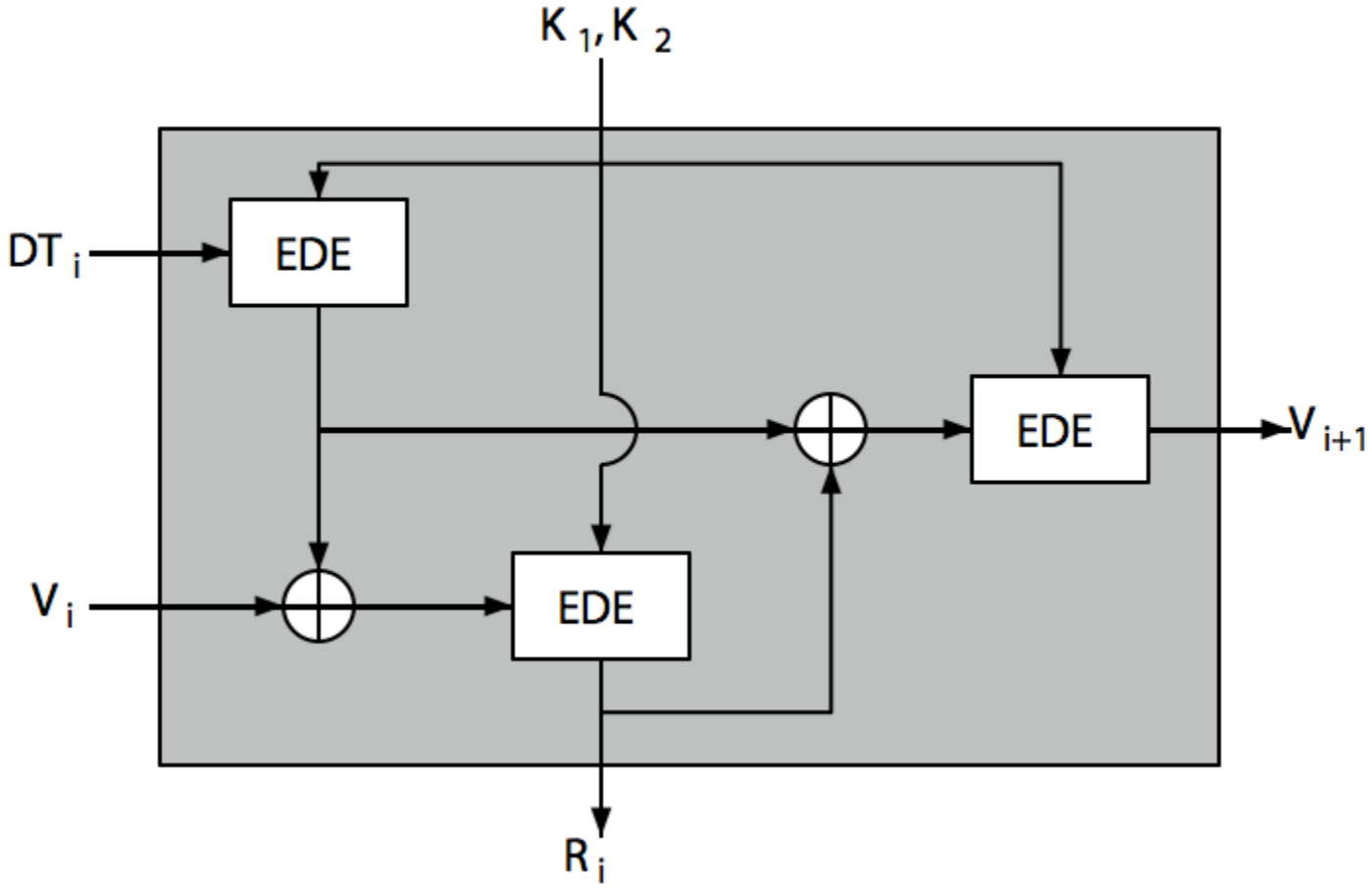
Using Block Ciphers as PRNGs

- for cryptographic applications, can use a block cipher to generate random numbers
- often for creating session keys from master key
- Counter Mode

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

  $X_i = E_{Km}[X_{i-1}]$

ANSI X9.17 PRG



Blum Blum Shub Generator

- Based on public key algorithms
- use least significant bit from iterative equation:
  - $x_i = x_{i-1}^2 \bmod n$
  - where $n=p.q$, and primes $p,q=3 \bmod 4$
- 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

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

Summary

- Have considered:
  - use and placement of symmetric encryption to protect confidentiality
  - need for good key distribution
  - use of trusted third party KDC's
  - random number generation issues

Hybrid Key Distribution

- retain use of private-key KDC
- shares secret master key with each user
- distributes session key using master key
- public-key used to distribute master keys
  - especially useful with widely distributed users
- rationale
  - performance
  - backward compatibility