Week 7

Lecture 1

Key Management

Lecture 2

MST

Workshop 7: Workshop based on Lectures in Week 6

Quiz 7

Key Management

COMP90043

Lecture 1

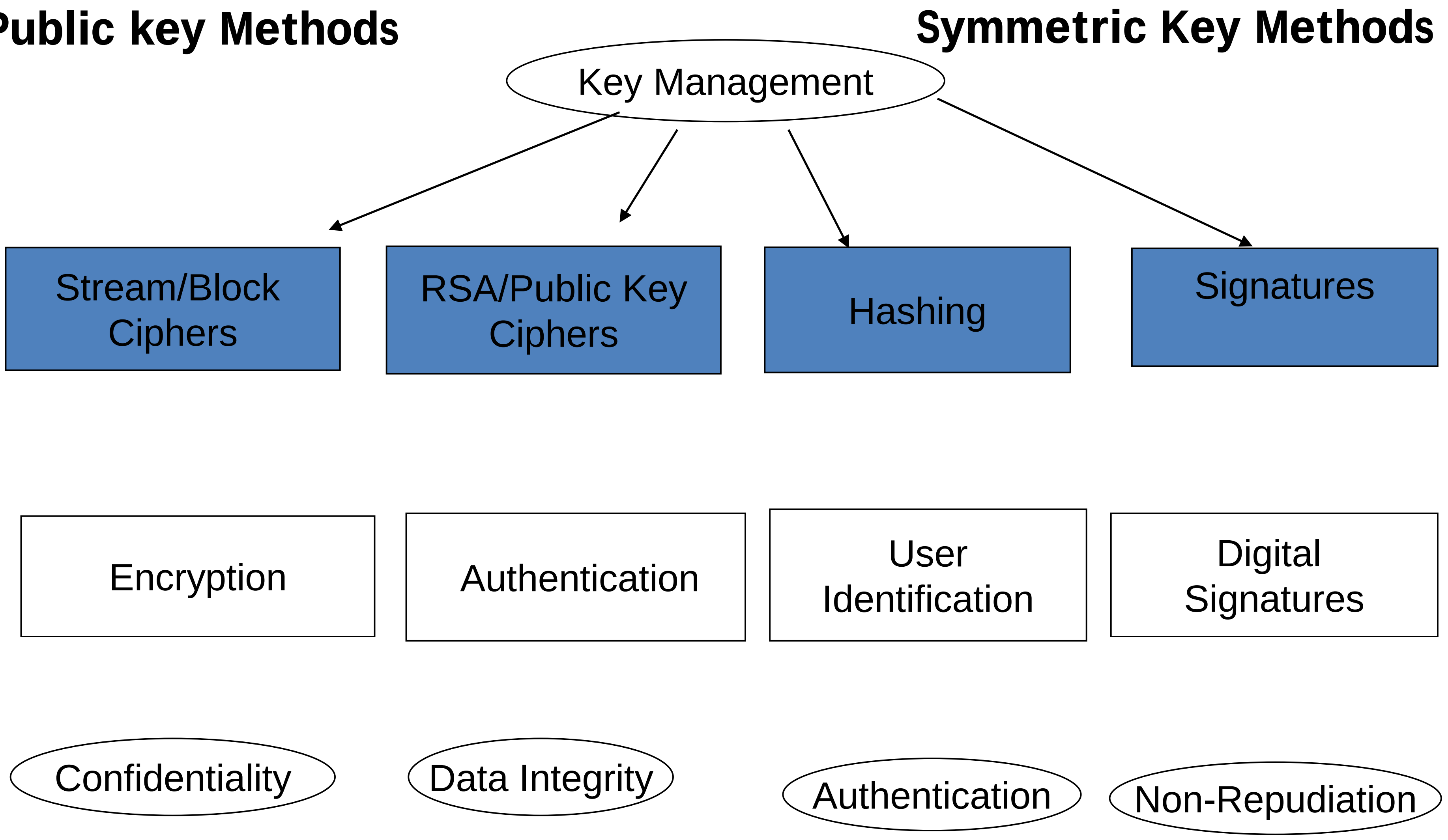
Lecture 1

- 1.1 Symmetric Key Distribution
  - Key Distribution using a key Server
  - Key Hierarchy
  - Key Distribution Scenario
  - Merkle Key Distribution
  - Merkle’s Scheme
  - Hybrid Scheme
- 1.2 Random Numbers
  - Pseudorandom Number Generators
  - Linear Congruential Generator
  - Using Block Ciphers as PRNGs
  - Blum Blum Shub Generator
  - Natural Random Noise

Key Management and Distribution

- Until now, we discussed many cryptographic primitives with different security objectives.
- When used in networked and computer systems, we need a plan to distribute appropriate keys to the relevant programs that use the primitives.
- This lecture will focus on symmetric key distribution methods.
- The issues are complex as it would involve multiple domains of engineering and computer science. These methods need to be included into protocols, operating systems etc.
- Most cryptographic algorithms except Hash functions involve keys.
- Key distribution is all about generation, distribution, storage, archival, …. of keys (symmetric and public).
- Without proper and secure key management, cryptographic algorithms are useless.

Key Management



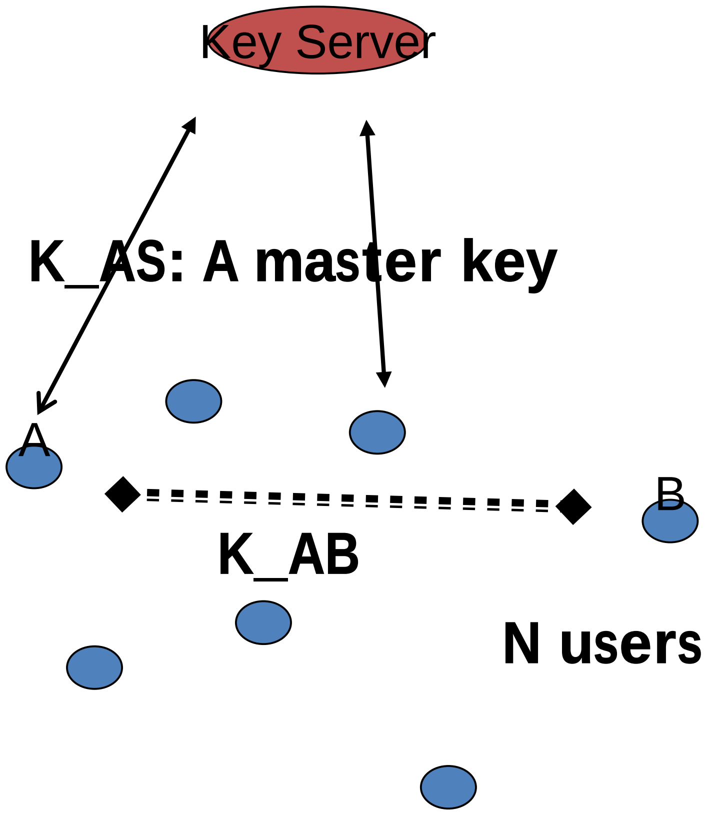
Keys in General

- Keys are to be formed using purely random sources, but in practice, they are usually pseudo random based on some secret seeds or random seeds obtained from physical means.
- Long life keys should be generated from a truly random source, examples: Thermal noise, time between key strokes etc.
- **Basic idea of Pseudo-random Bit Generators:**
- **Extend truly generated random number of length k to length t, where t > k.**

Symmetric Key Distribution

- Consider symmetric key encryption system involving Alice and Bob.
- We studied that the security of Symmetric key systems is based on the following requirements:
  – The encryption algorithm does not have any weakness
  – The secret key is private to the sender and receiver.
- What are the methods of key distribution?
- The users can meet in advance, but this is impracticable as the users are heterogenous in communication networks.
- So we need some sort of key distribution framework as the keys need to be private between users and their access must be denied to others in the network.
- Also the keys need to be frequently changed to minimize the risk of attack.
- In practice, failures in secure system are result of vulnerabilities in key management rather than the weaknesses of encryption functions.

Symmetric Key Distribution



- Main Question:
- A and B do not share a secret channel.
- How to distribute a secret session key K_AB to parties A and B?
-
- Requirement:
- A new session key is needed for each new session and for each new security function
- (authentication, data confidentiality, integrity)

Different Methods

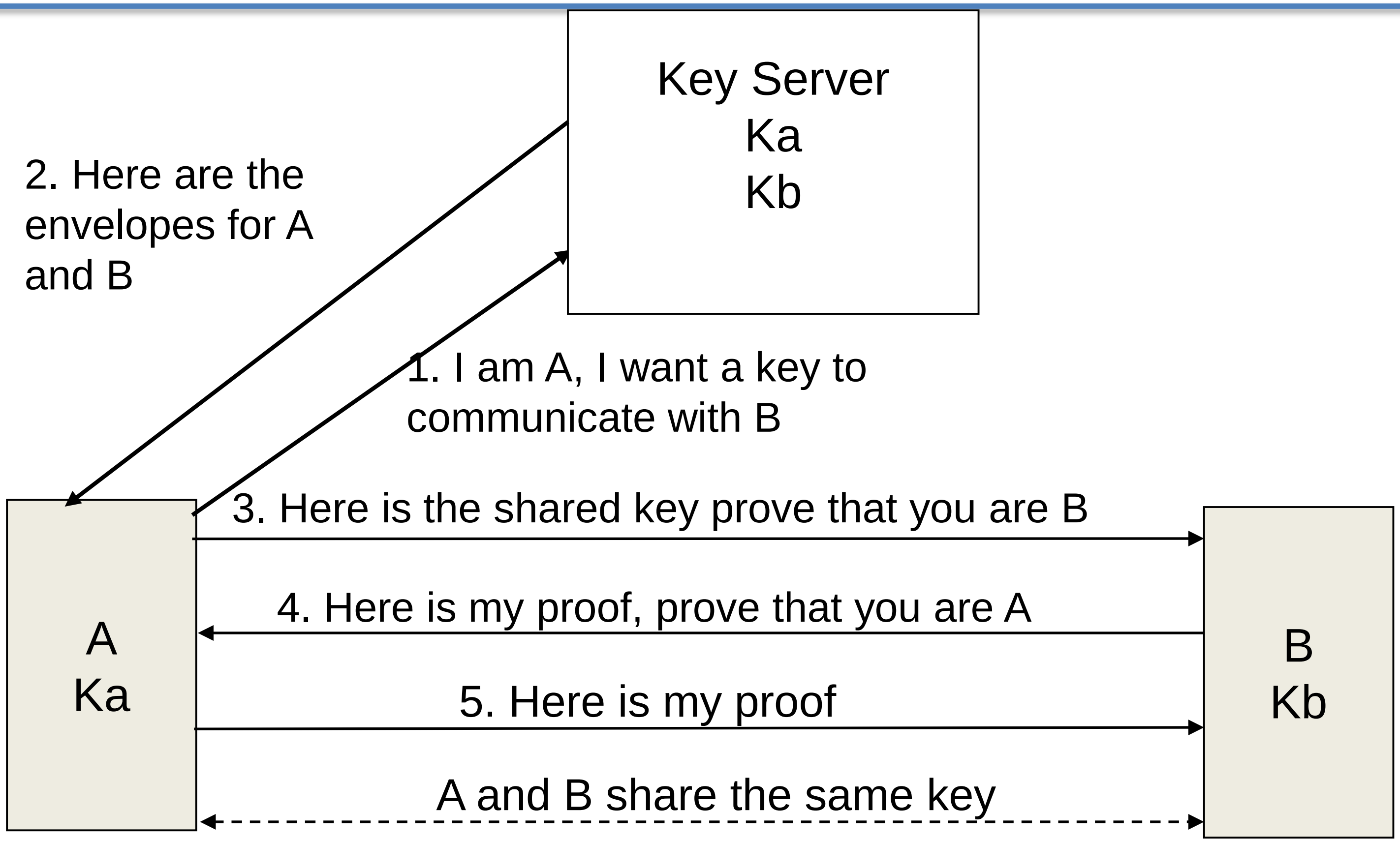
- N users
- Direct sharing between nodes : Complexity of initial key installation : O(N^2)
- Public Key Algorithmic Method (Diffie-Hellman Protocols or using RSA) : O(N)
- Using a Key server (O(N))

Methods of Key Distribution

- Stallings lists the following four alternatives:

1. A can select key and physically deliver to B
2. A 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 using a key Server


Key Server
Ka
Kb
2. Here are the envelopes for A and B
1. I am A, I want a key to communicate with B
3. Here is the shared key prove that you are B
4. Here is my proof, prove that you are A
5. Here is my proof
A and B share the same key
A
Ka
B
Kb

A and B need to trust the key server:
- security of session keys
- entity authentication

Server Based protocols

•Needham-Schroeder Protocol

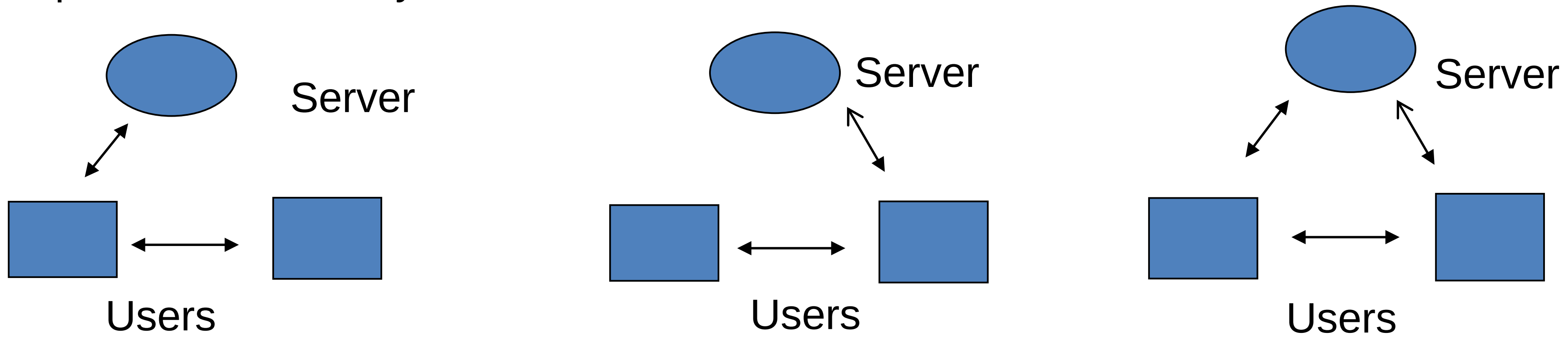
Problems with freshness of session keys

•Otway-Rees Protocol

Use active authentication to avoid problems with using old keys

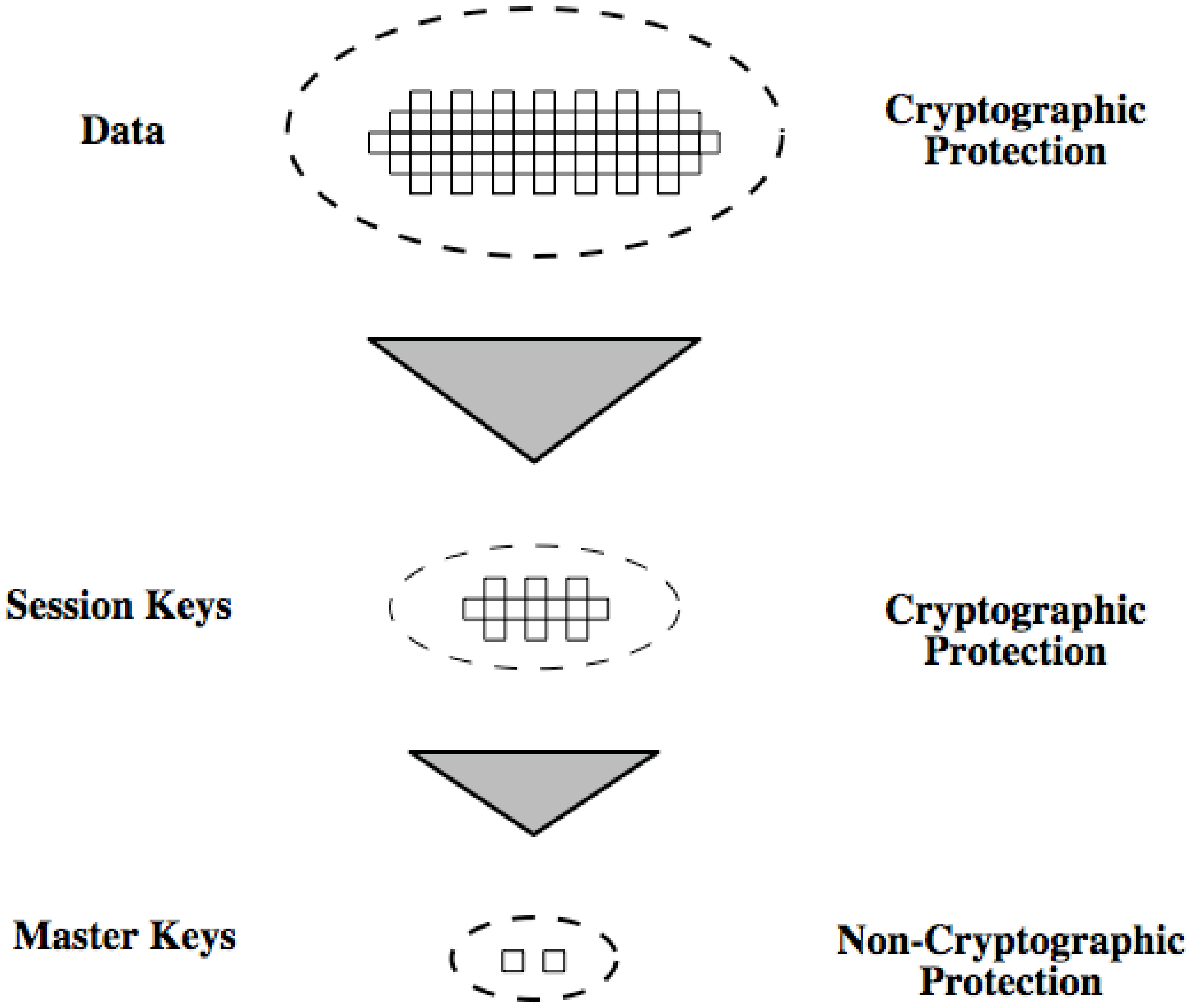
We will only follow the protocols given in the textbook.

Handbook of Applied Cryptography by Menezes et. Al. gives different possibilities of Key Server and user interactions:



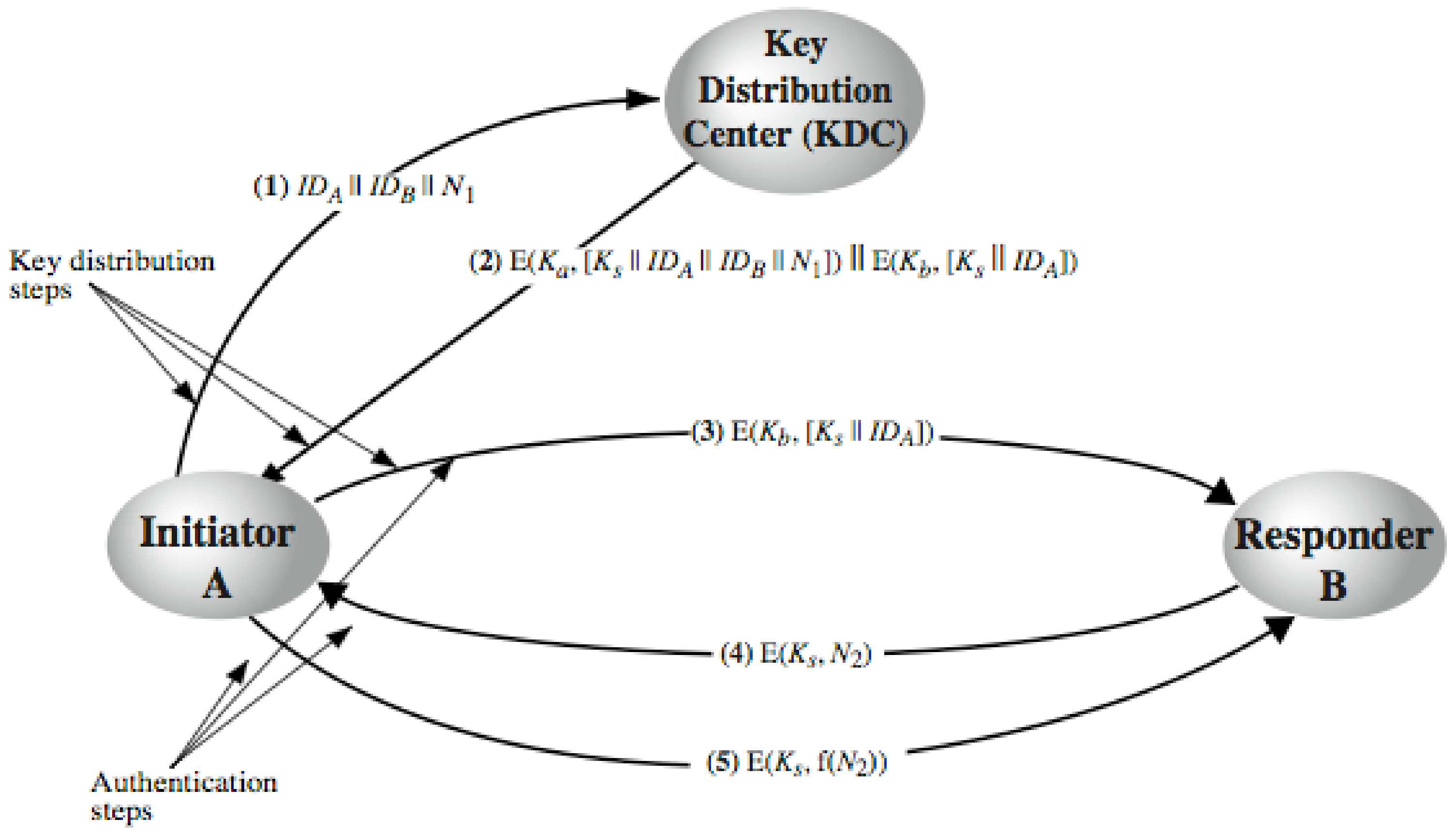
Key Hierarchy

- **Session key**
  - temporary key
  - used for encryptic of data between users
  - for one logical session then discarded
- **Master key**
  - used to encrypt session keys
  - shared by user & key distribution centre



From the textbook Fig 14.2

Key Distribution Scenario


Key Distribution Center (KDC)
(1) $ID_A \| ID_B \| N_1$
Key distribution steps
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
Initiator A
Responder B
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

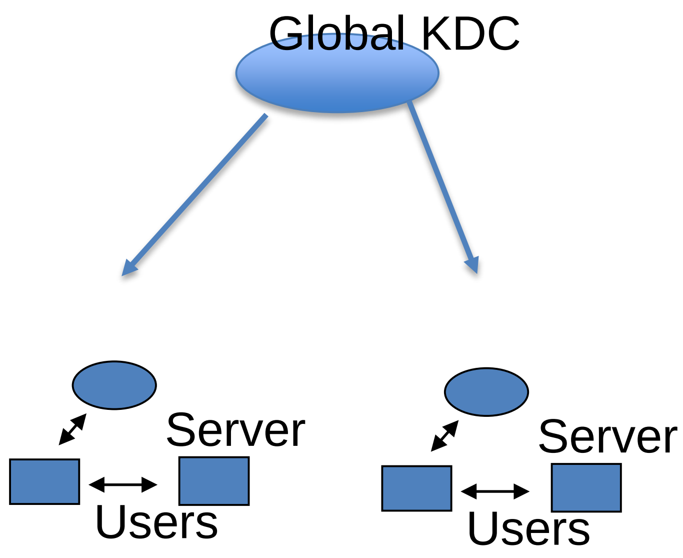
This is a version of Needham-Schroeder Protocol
used by Stallings

From the textbook a version of Fig 14.3

Hierarchical Key Control

- Key distribution method can be extended to multiple KDCs, a local KDC and a global KDC.
- You can have a hierarchy of KDCs.
- Users within a same local domain are supported by the local server,
- Users in two different domain will need involve global KDC to exchange keys.

- Hierarchy of keys minimizes complexity of key distribution. Also localizes the risk of fault or compromise within a local domain.



Session Key Lifetime

- How often session keys should be changed?
- There are two considerations: frequent changes
  - increases security
  - but adds delay to the communication and hence reduces network capacity.

- The textbook explains issues with respect to connection oriented protocols in detail, please read Section 14.1

- Key control is an interesting practical topic.

Decentralized Key Control

- In the previous method we need to trust KDC which needs a protection from being compromised.
- A fully decentralized approach may require every node establish a master key with every other node, thus needs n(n-1)/2 keys for n end point system.
- A session key algorithm is explained in Fig 14.5 of the textbook.

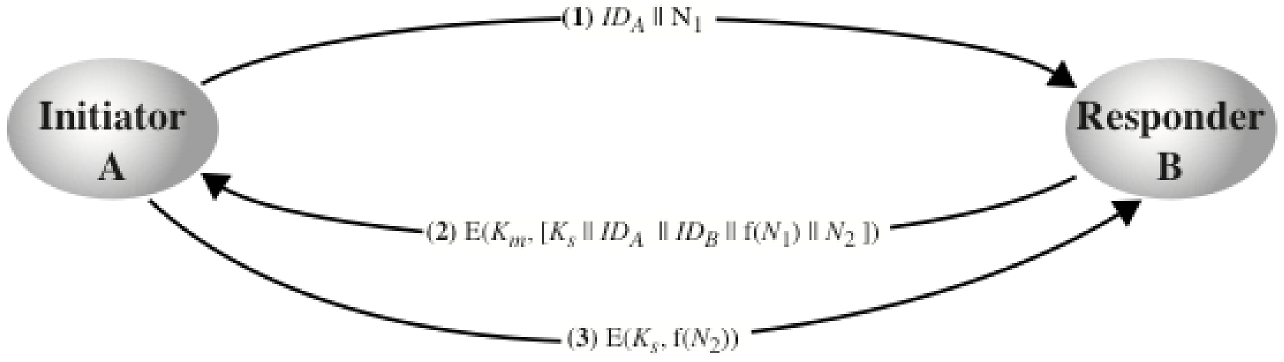


Figure 14.5 Decentralized Key Distribution

From the textbook a version of Fig 14.5

Controlling Key Usage

- Having some control on the nature of keys will help to manage them effectively.
- It may help to identify different types of keys:
  - Data Encrypting keys (DEK) for general communication across networks
  - PIN-encrypting Key for PINS and Electronic transfer applications
  - File Encrypting Key, for accessing files in public locations.

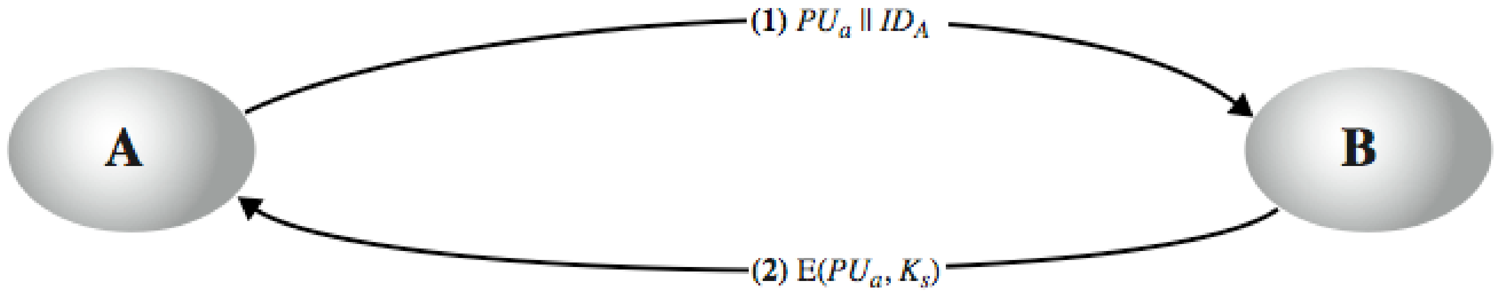
- The textbook suggests a method based on DES keys using 8 reserved bits, but tag length is limited.
- A more flexible scheme is depicted in Fig 14.6 of the textbook.

Using Asymmetric Encryption

- Public key was introduced with a promise to reduce the key usage.
- Can we use public key system to exchange session keys?
- Yes, you could do. But only problem is that the end points need to have the authentic public key of the other node.
- Also, in general public key encryptions are slower than symmetric key counterparts and hence they are not used for direct encryption. In general, always a hybrid scheme is employed-first use public key to obtain a session key and then use the session key in a symmetric encryption for efficiency.
- Let us consider a simple idea due to Merke.

Merkle Key Distribution

1. A generates a public/private key pair [*PUa*, *Pra]* and transmits a message to B consisting of PUa and an identifier of [A, IDA]
2. B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
3. A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
4. A discards PUa and PRa and B discards PUa.



From the textbook a version of Fig 14.7

Merkle’s Scheme

- Was the previous Scheme Secure?
- Main issue is authentication of suer’s public key, vulnerable to Main in the Middle Attack.
- To stop such attacks, users need a method for authentication of public keys.

From the textbook a version of Fig 14.8

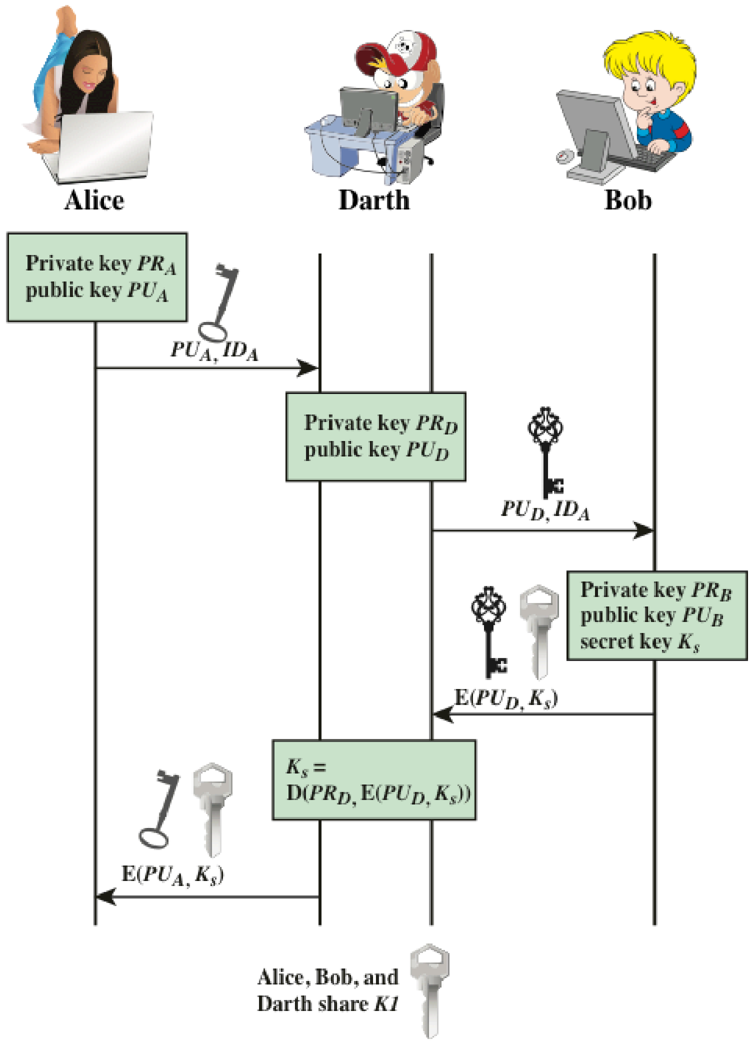

Alice
Darth
Bob
Private key PR_A public key PU_A
PU_A, ID_A
Private key PR_D public key PU_D
PU_D, ID_A
Private key PR_B public key PU_B secret key K_s
$E(PU_D, K_s)$
$K_s = D(PR_D, E(PU_D, K_s))$
$E(PU_A, K_s)$
Alice, Bob, and Darth share $K1$

Figure 14.8 Another Man-in-the-Middle Attack

Hybrid Scheme

- IBM main frames use this method based on public key.
- The scheme retains a KDC that shares a master key with each user.
- A public key scheme is used to distribute the master keys.
- The consideration is mainly on performance and backward compatibility.

Random Numbers

- Where **random numbers** are used in cryptography?
  - Session keys
  - Authentication protocols to prevent replay
  - Public key generation-SSL
  - Keystream for a one-time pad
- What are the properties you are looking for in these random numbers?
  - Having Uniform distribution, Statistically random, independent
  - Unpredictability of future values from previous values

Pseudorandom Number Generators (PRNGs)

- Deterministic algorithm techniques of generating random like objects from a seed or another random source.
- They are truly random, but satisfies many tests for statistical randomness.
- They are thus called Pseudorandom numbers.

Linear Congruential Generator

- Most common method used in practice, part of many OS implementations.
- The sequences follow a linear relation:

  $X_{n+1} = (aX_\mathrm{n} + c) \bmod m,$

  Where X_0 is the seed and a and c are constants.

- If choose appropriate a and c and X_0, you obtain a long sequence which looks random.

- Very useful in practice- Main criteria:
  - function generates a full-period
  - generated sequence should appear random
  - efficient implementation with 32-bit arithmetic
- The method is not secure since an attacker can reconstruct sequence given a small number of values.

Using Block Ciphers as PRNGs

- Stallings discussed many techniques based on block ciphers to generate random numbers.

- Popular ones:

- Counter Mode

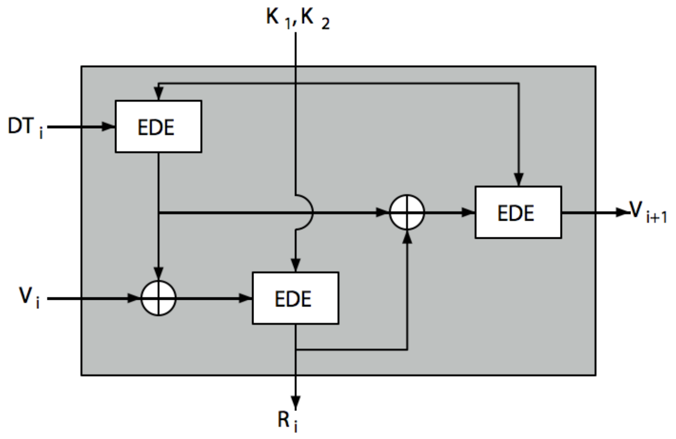
  $X_i = \mathrm{E}_{Km}[\mathrm{i}]$,

- Output Feedback Mode

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

ANSI X9.17 PRG

- DTi - Date/time value at the beginning of ith generation stage
- Vi - Seed value at the beginning of ith generation stage
- Ri - Pseudorandom number produced by the ith generation stage
- K1, K2 - DES keys used for each stage
- Then compute successive values as:
- Ri = EDE([K1, K2], [Vi XOR EDE([K1, K2], DTi)])
- Vi+1 = EDE([K1, K2], [Ri XOR EDE([K1, K2], DTi)])



From the textbook

Blum Blum Shub Generator

- This is an interesting generator based on public key cryptography.
- You start with a seed mod n and the sequence are obtained as:
- x_i = LSB(x_{i-1}^2) mod n, where n=p.q, and primes p,q=3 mod 4, LSB: Least Significant Bit.
- Interestingly this algorithm has high level of security,
- If someone can break the “next-bit” test, then you can use him to break the factorization of n, which is considered hard-so it hs strongest public proof of its strength.
- However it is very slow.

Natural Random Noise

- One can use known natural randomness in real-world to your advantage.
- There are many events which look random: adiation counters, radio noise, audio noise, thermal noise in diodes, leaky capacitors, mercury discharge tubes etc.
- Please read Section 8.6 of the textbook.

Week 7

Lecture 1

Key Management

Lecture 2

MST

Workshop 7: Workshop based on Lectures in Week 6

Quiz 7