DI-FCT-UNL
Segurança de Redes e Sistemas de Computadores
Network and Computer Systems Security

Mestrado Integrado em Engenharia Informática
MSc Course: Informatics Engineering
1º Semestre, 2019/2020

- Key Distribution Protocols using (only) Symmetric Encryption
- Authentication in the Kerberos System and Protocol

Symmetric Cryptography: Shared Secret-Key Cryptography

Use of Symmetric Cryptography to build Secure Channels

- Remember ref. on OSI X.800 framework for security properties, security services, security mechanisms and attack typology
- **Symmetric cryptograpghy: target is on Confidentiality**
- ***CMAC Constructions and specific modes (ex., GCM, CCM), target also on Integrity and Message Authentication Codes***

- But no Peer-Authentication and No Repudiation Guarantees
    - Does not protect sender from receiver forging a message & claiming is sent by sender (or vice versa)

Symmetric Cryptography: Shared Secret-Key Cryptography

- Need to distribute, establish and manage keys in a secure way
  - If shared keys are disclosed communications will be compromised (*NDA** of keys between principals involved)
  - For security keys can be established for short periods:
    - Session Keys (as temporary or short-term keys)
    - Need rekeying: fast and secure !

Secure Key Distribution Protocols

NDA – Non Disclosure Agreement

Outline Today

- **Key (or security association parameters) distribution using symmetric encryption**

  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Outline Today

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Key Distribution Options

- A key could be generated and selected by A and physically delivered to B.
- A third party (or KDC) could generate and select the key and physically deliver it to A and B.

Physical (or manual) delivery

- If A and B have previously and recently used a key, one party could transmit the new key to the other, using the old key to encrypt the new key.

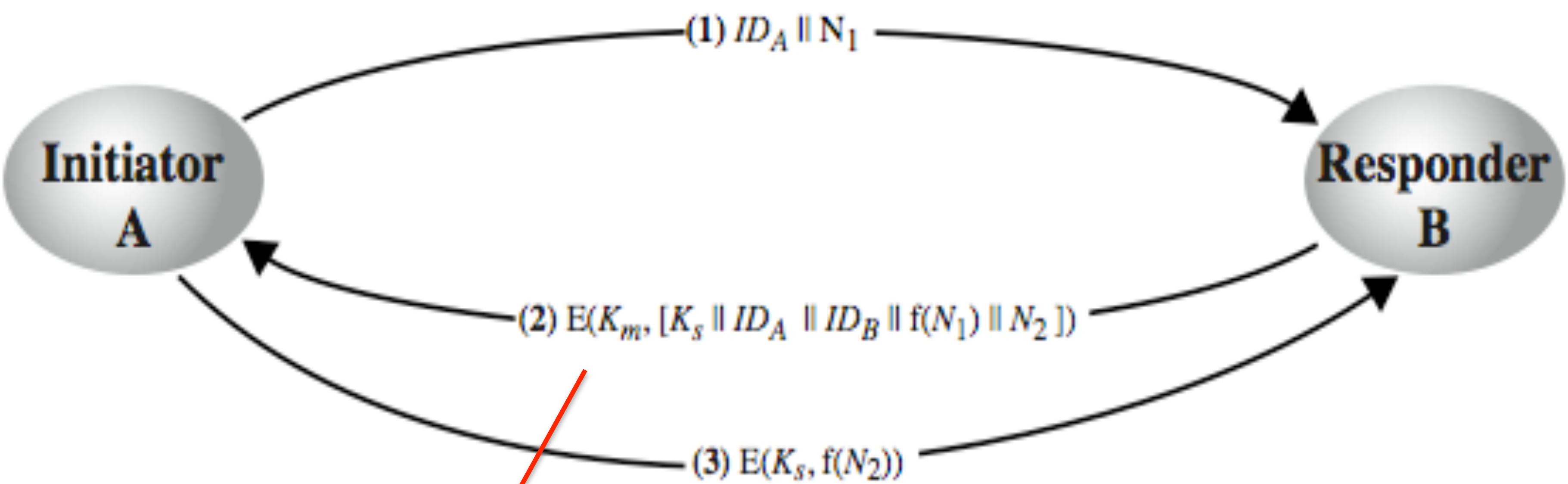
Link or End-to-End

- If A and B each have an encrypted connection (shared master key) to a third party KDC, KDC could deliver a key on encrypted links to A and B.

Master (or Permanent Key) and established Session (temporary) Keys

Discussion: decentralized control

- Peer to Peer Key Distribution, using pre-shared Master Keys (long term) to generate Session Keys (short term)



Session Key (Ks) distributed Encrypted with a Pre-Shared Master Key Km)

Peer-Rekeying from a previously shared key

If "A" and "B" have a previous shared key, one party can transmit a new generated session key to the other, encrypting it with the old key.

- Problems ?
  - **No Perfect Forward Secrecy (PFS) and Perfect Backward Secrecy (PBS)** guarantees
  - **No Key-Independence**
  - **Not scalable**: possible millions of keys still must be generated and distributed in the environment
  - What about the "**key-process quality generation control**", from the viewpoint of each principal

Key-management and the scale problem

- For link-encryption or point-point
  - One key for each pair of hosts on the network that wish to communicate
    - 1 key for 2 hosts, 3 keys for 3 hosts, 6 keys for 4 hosts, …

[N (N-1)] / 2 keys for N hosts

- **Half-million keys for 1000 nodes**

Scale implies on: key-management control problem with non-disclosure security guarantees. Realistic ? Secure ?

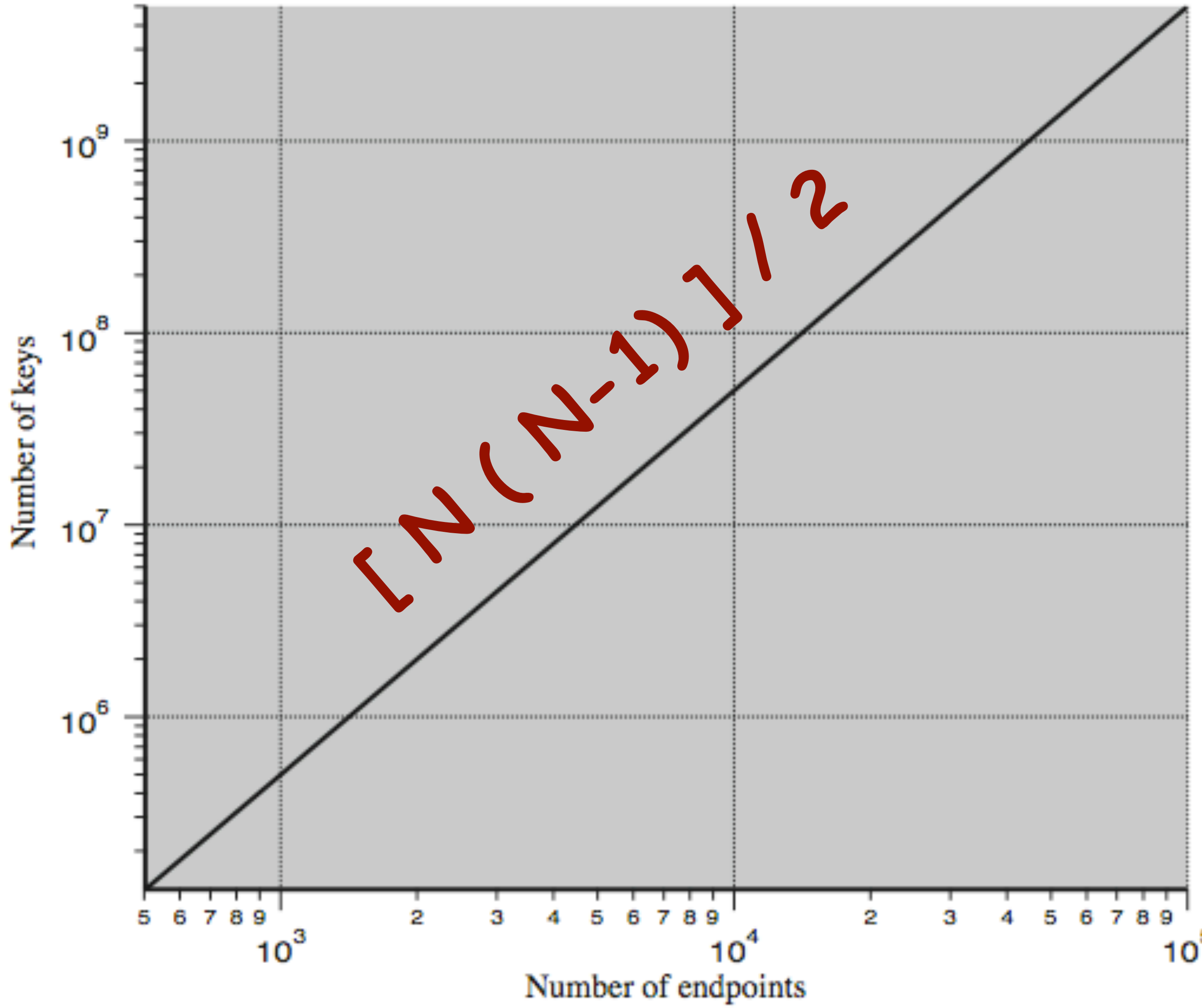
- For “end-to-end” encryption ?

  Example 1000 nodes, 10000 applications

  - **50 x 10^6 keys**

The scale problem: N principals (or peers)

Complexity:
N^2



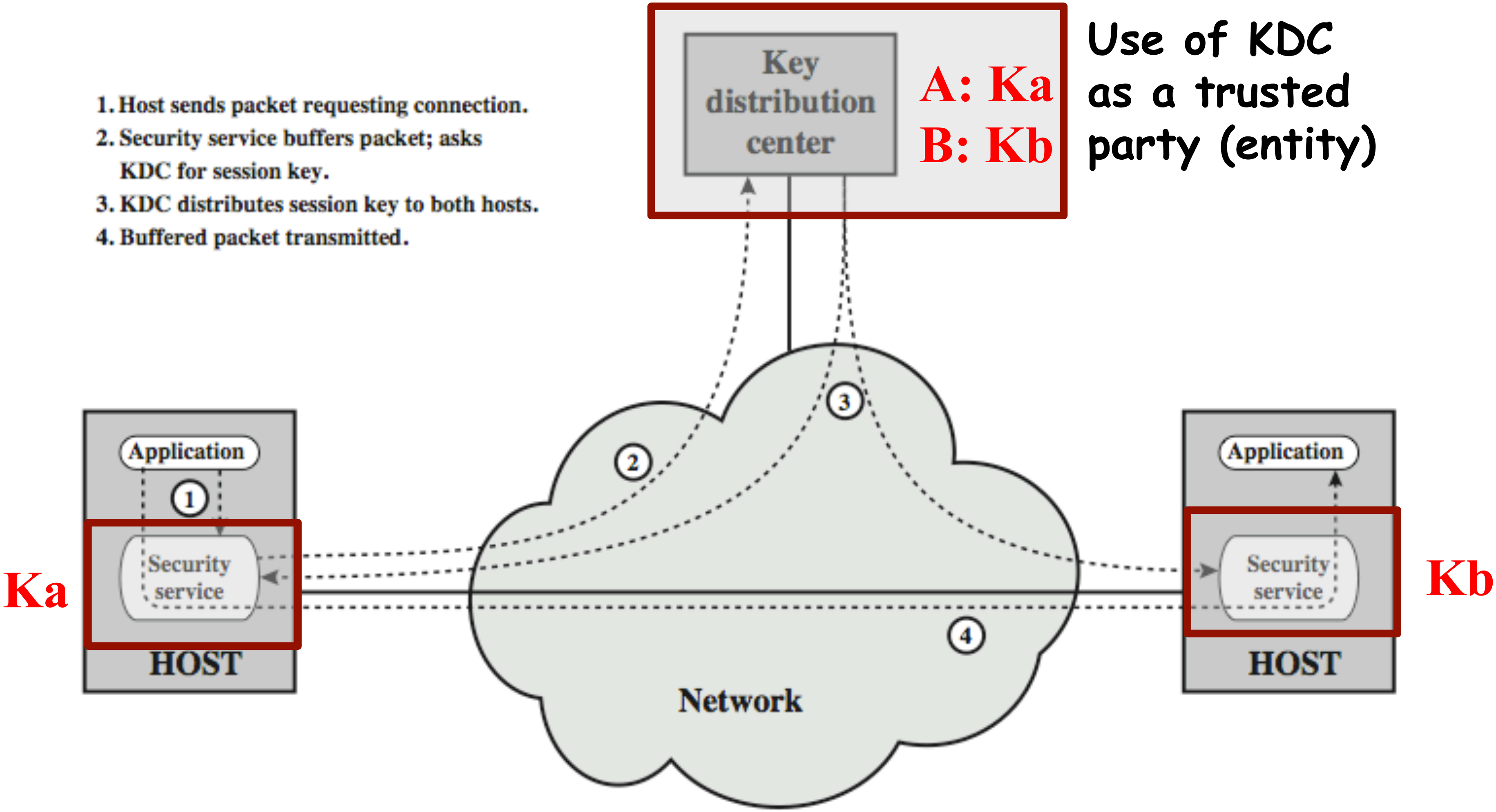
Better Solution ?

Outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

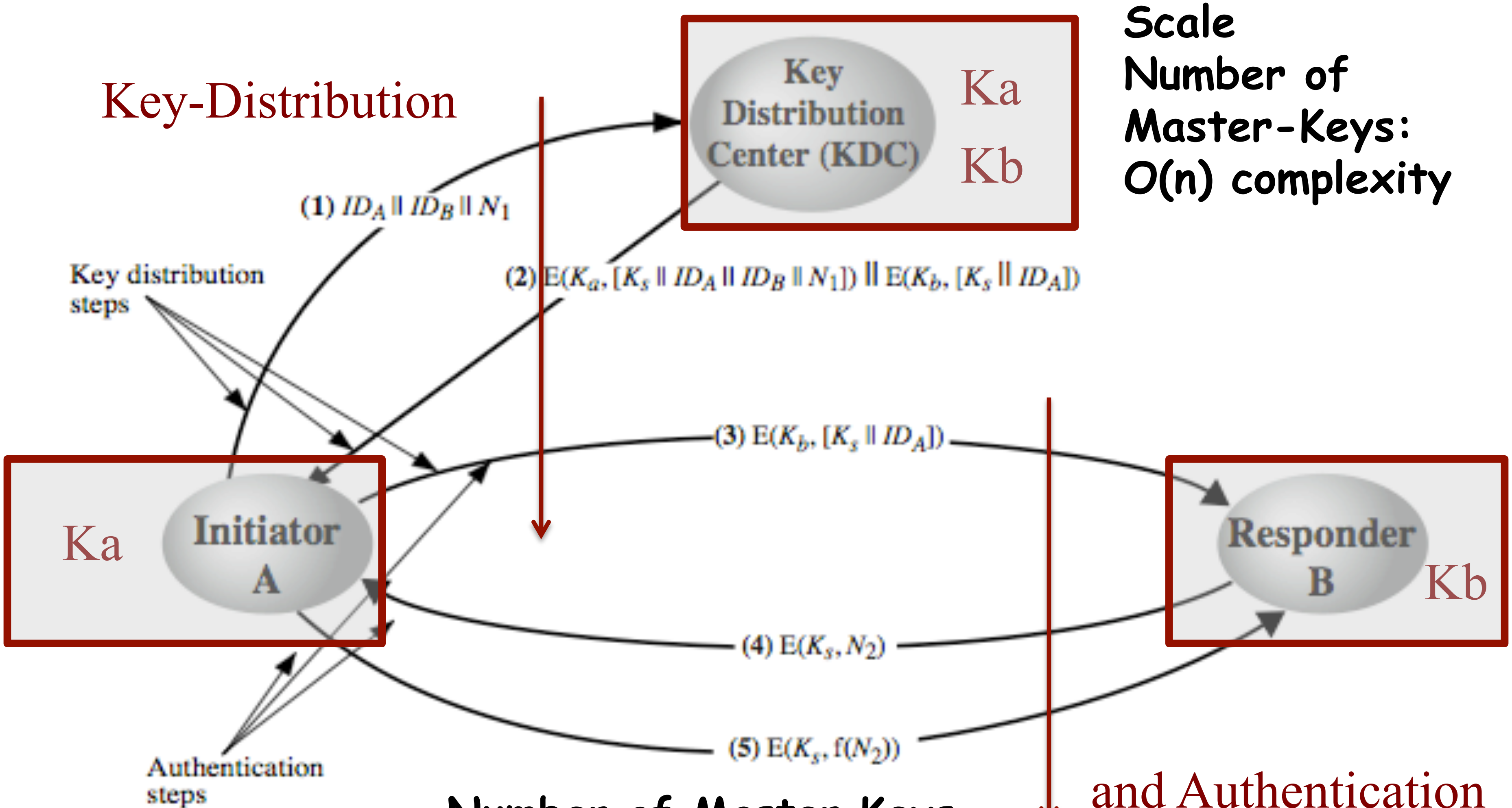
Solution: use of Key-Distribution Centers (KDCs)

1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.



Use of KDC as a trusted party (entity)

Key-Distribution Scenario with a KDC



Scale
Number of Master-Keys: O(n) complexity

Number of Master Keys on principals: 1 Master Key per KDC

Better scale: hierarchical models

- Hierarchical key control: possible implementation of more than one level for KDCs
  - Good for scalability, load-balancing, avoidance of single points of failures/attacks
  - A KDC in a hierarchy may be responsible for a more "small" domain (ex., a LAN, a LAN segment, etc)
  - Local domain KDCs can ask for a key to a KDC in the next layer of the hierarchy

  - Can address secure communication (secure channels) between principals in different domain (cross-domain security channels):
    - Local domain KDCs can ask for a key to a KDC in a different domain

Discussion: lifetime of session-keys

Master keys (long term) vs. Session keys (short term)

- More **frequent "rekeying"** (or **fresh session keys**) => more security
  - "Hard" for brute-force or cryptanalysis attacks
- But **rekeying => overhead**
  - More latency, network-traffic burden, synchronization of keys, …
  - Need fast rekeying mechanisms !
- **Flexibility: choices for different options:**
  - **Connection-oriented communication**
    - Can map: one session key for one connection
    - What if connections are "long" ?
  - **Connectionless communication**
    - What is the "session" for a key-session in this case ?

Rekeying strategies

Different choices with possible different criteria:

- Ex., rekeying in each PDU sequence number cycle
  - Requires sync. counters and reliable delivering
- Ex., temporal "rekeying"
  - Requires time synchronization
- Ex., Random-based rekeying
  - Requires the initiation of a synchronization protocol
- Ex., Event-based rekeying
  - Requires the synchronization of such events
- .....

Outline

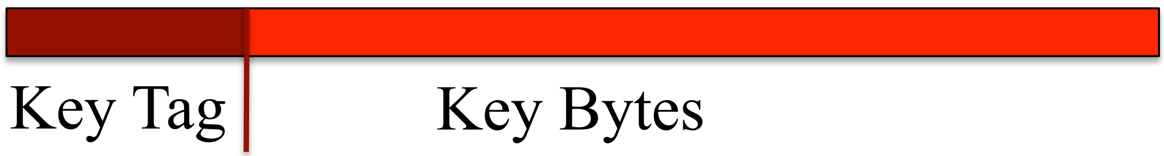
- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Key-Management Issues: Keystores

In practice we must use secure keystores (or key rings): separation control between "master-keys" and "session keys" (in the keystore)

- Usually need different types of keys
    - Data or Message Encryption Keys (for different protocols, different uses)
    - PIN encrypting keys for different personal PINs
    - File-Encryption keys for different applications, …
    - CMAC or HMAC keys …
- Key-usage controls: ex., reservation of some "key bits" or added "key-selector fields", as usage-control tags

Key Tags used as Key-Selectors



Ex., 8 bit TAG: XXXXXXXX

1 Master Key
0 Session Key

1 Encryption Key
0 Decryption Key

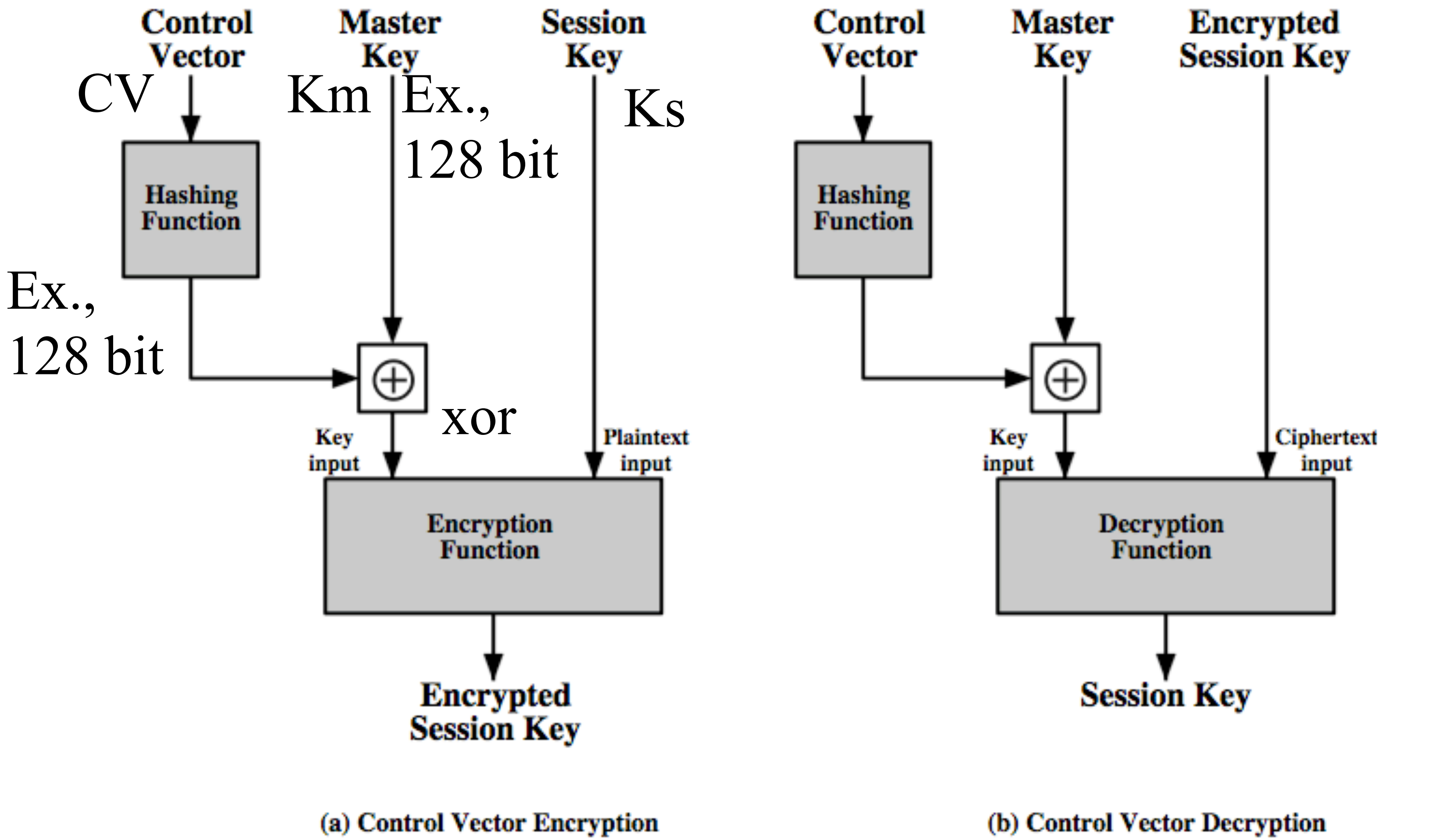
Application ID Selector
Ex., 4 bits, 16 Applications

2 bits (additional)
Other purposes

Keystores with control vectors

- Each key has an associated control vector of a variable size (used as the tag selector)
- Control vector is cryptographically coupled with the key in "key-generation" time, in the KDC
  - Standardized "coupling and decoupling processes"

Control Vector Encryption / Decryption



Ks delivering form KDC: CV || Encrypted Session Key

CV can be splitted in TAG (selector) and Counter/Salt

Other Techniques and Technology

Password-Based Encryption

- See in Practical class
- Can use hash of PWDs combines with PBEncryption, to distributed encrypted session keys"

Key-Wrapping Constructions and Techniques

- We can see these constructions in the Lab
- Wrapping keys (as master keys) used for secure envelope constructions encrypting session keys
- Can combine onion-encryption and different encryption algorithms

Use of secure physical keystores: ex., security dongles

Better: Smart dongles, Smartcards, HSMs

Outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

KDPs: Summary of Concerns

- **Protocol efficiency**
  - Computational vs. communication efficiency
  - Key refreshment vs. Rekeying efficiency
- **Fast and Secure Rekeying Guarantees**
  - Forward secrecy (FS) and Backward secrecy (BS)
  - Key-independence (KI)
  - Perfect secrecy conditions: FS+KI = PFS and BS+KI = PBS
- **Key-generation control**
  - Key-quality conditions
  - Contributive conditions
  - Fairness conditions
- **Formal security verification** (formal verification: model checkers, theorem proving, logic-analysis, complexity-theoretic proofs, security analysis)

Authenticated Key-Distribution

- When authentication guarantees are provided in the key-distribution protocol
- Sometimes, lack of authentication (or vulnerabilities against authentication attacks) are subtle, sometimes difficult to detect.

  - Ex.,

A>B	N_A	
B>A	**MAC_{KAB} (B, A, N_A), N_B**	**=> A: B is B**
A>B	**MAC_{KAB}(A, B, N_B)**	**=> B: A is A**

Reflection attacks

- Ex, Reflection by an adversary C.

C_A>B N_C

B >C_A MAC_{KAB} (B, A, N_C), N_B

C_B>A N_B

A >C_B MAC_{KAB} (A, B, N_B), N_A'

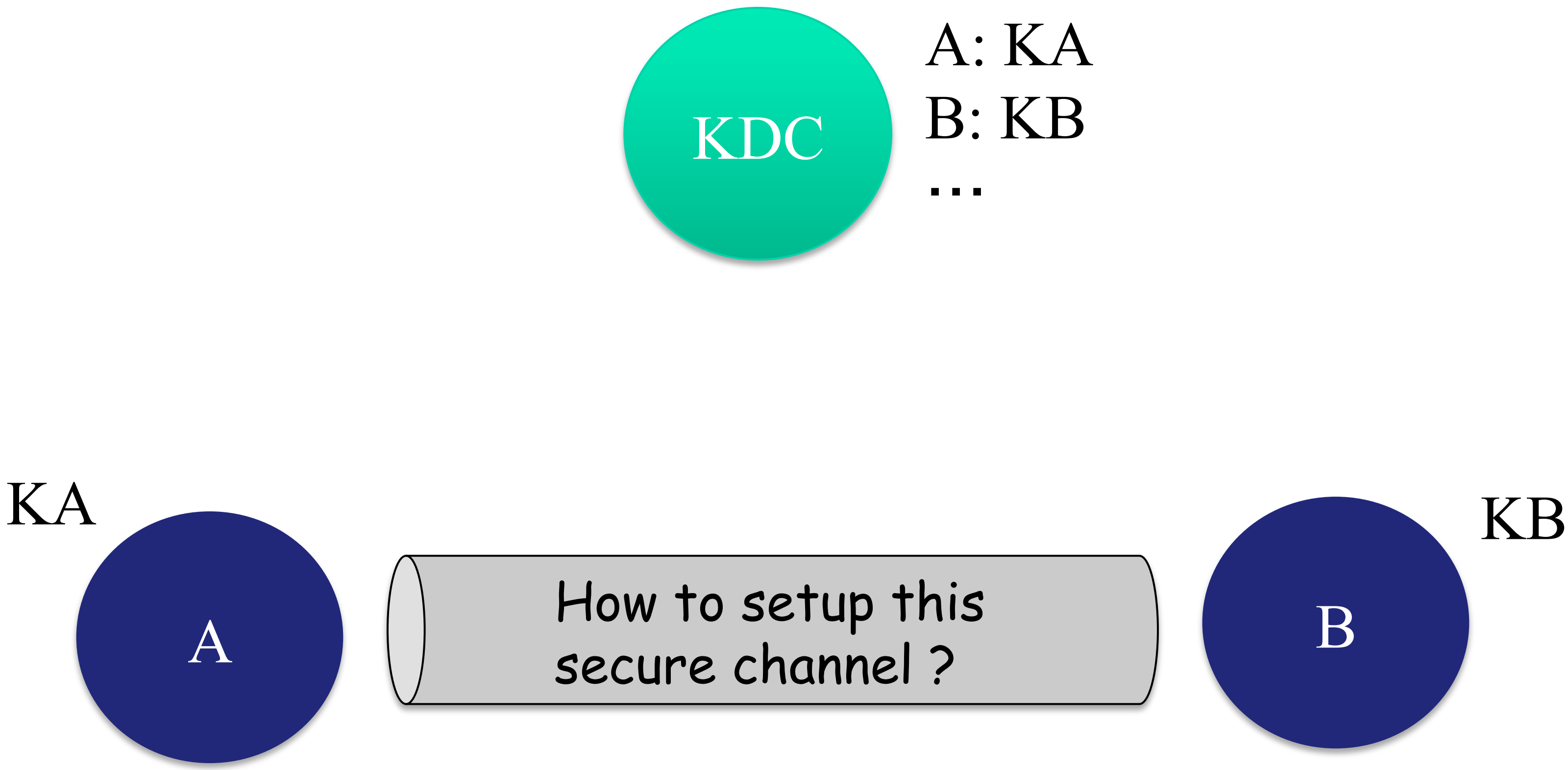
C_A>B N_A'

C_B>A MAC_{KAB} (A, B, N_B) => A: C_B is B

B >C_A MAC_{KAB} (B, A, N_A') => B: C_A is A

Key Distribution Protocols via KDC

Only Using Symmetric Cryptography and Symmetric Keys



KDPs using KDCs

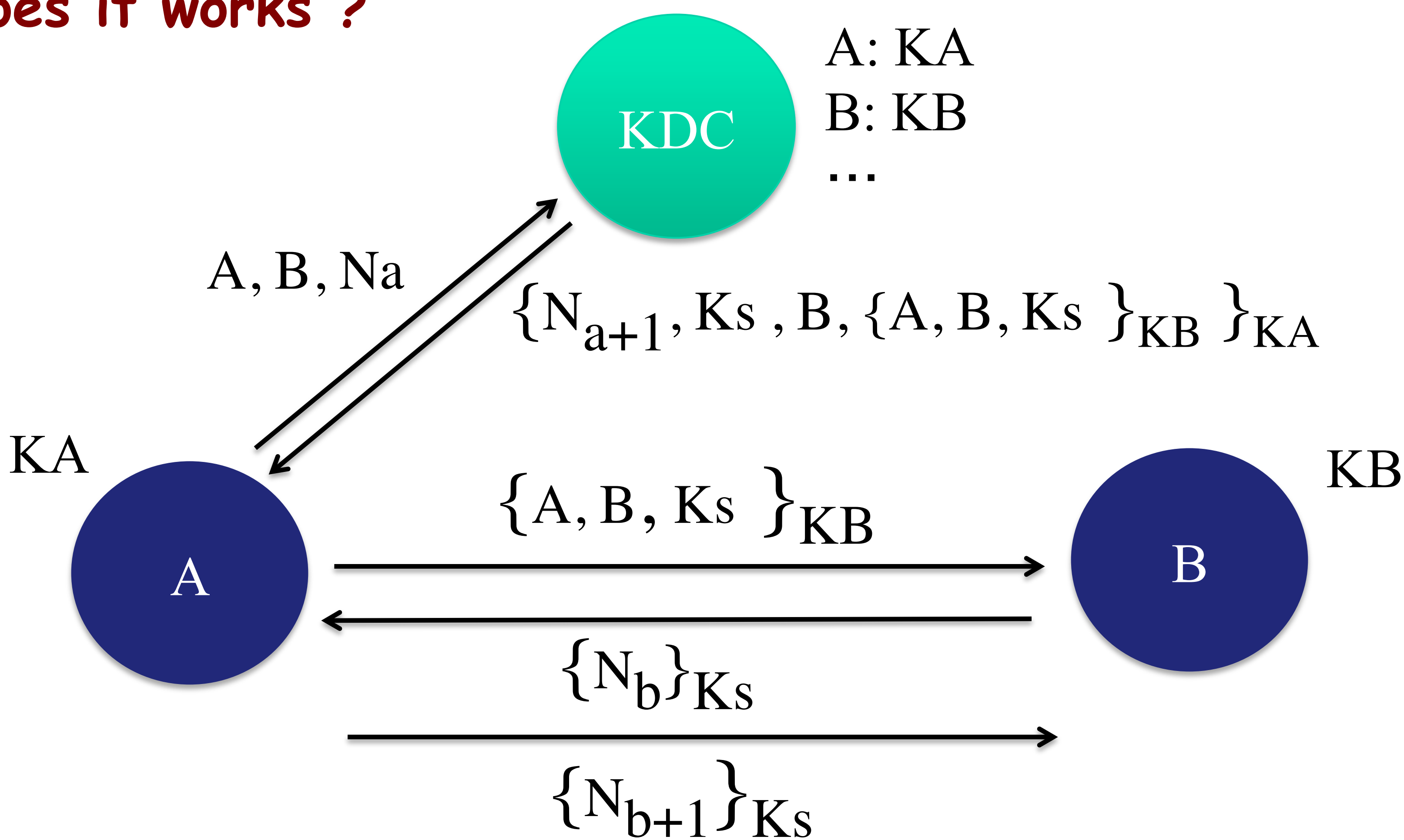
KDCs: Trusted Arbitration Entities

- All principals that want to establish mutual secure channels, need to share a master key (long time duration) with the KDC
- Many models of protocols proposed for a KDC-model
- Some Base Models:
  - Needham-Schroder (Symmetric Encryption) model
  - Otway-Rees
  - Yahalom
  - Wide-Mouth Frog
  - Neuman-Stubblebine

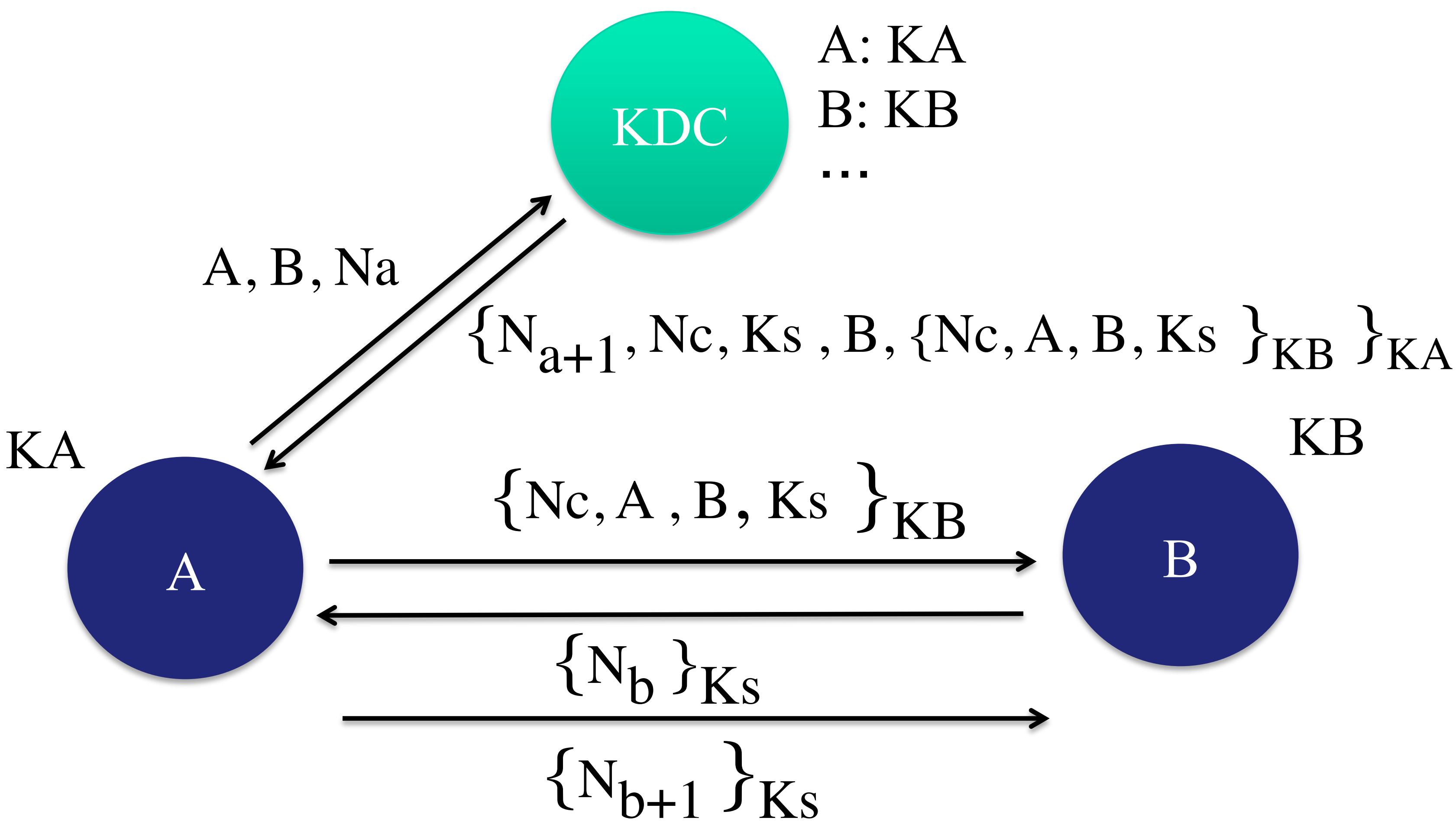
Disclaimer: will discuss these protocols focusing only on confidentiality assumptions … (easy to generalize for integrity and message-authentication guarantees)

Needham-Schroeder Protocol Model

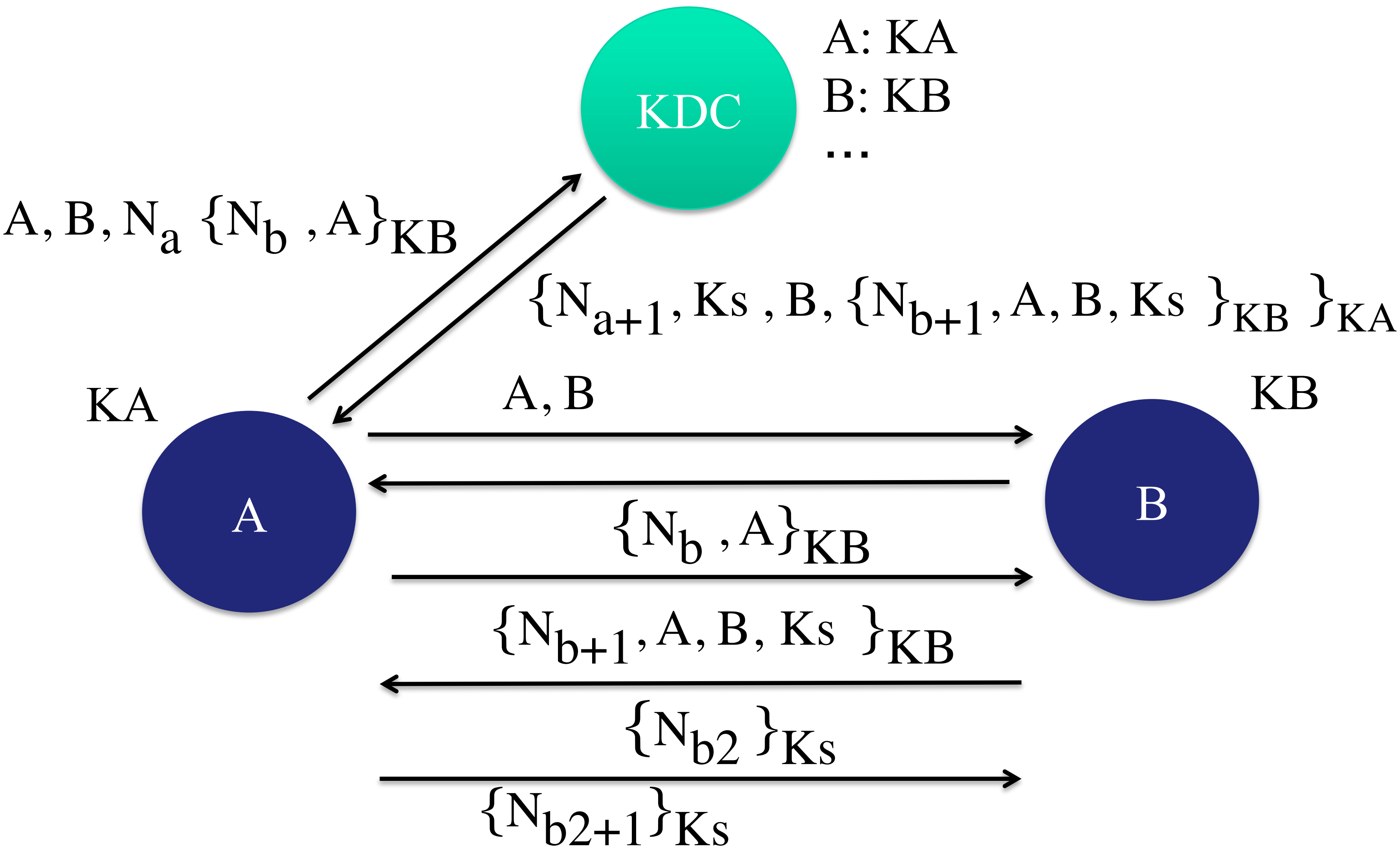
Does it works ?



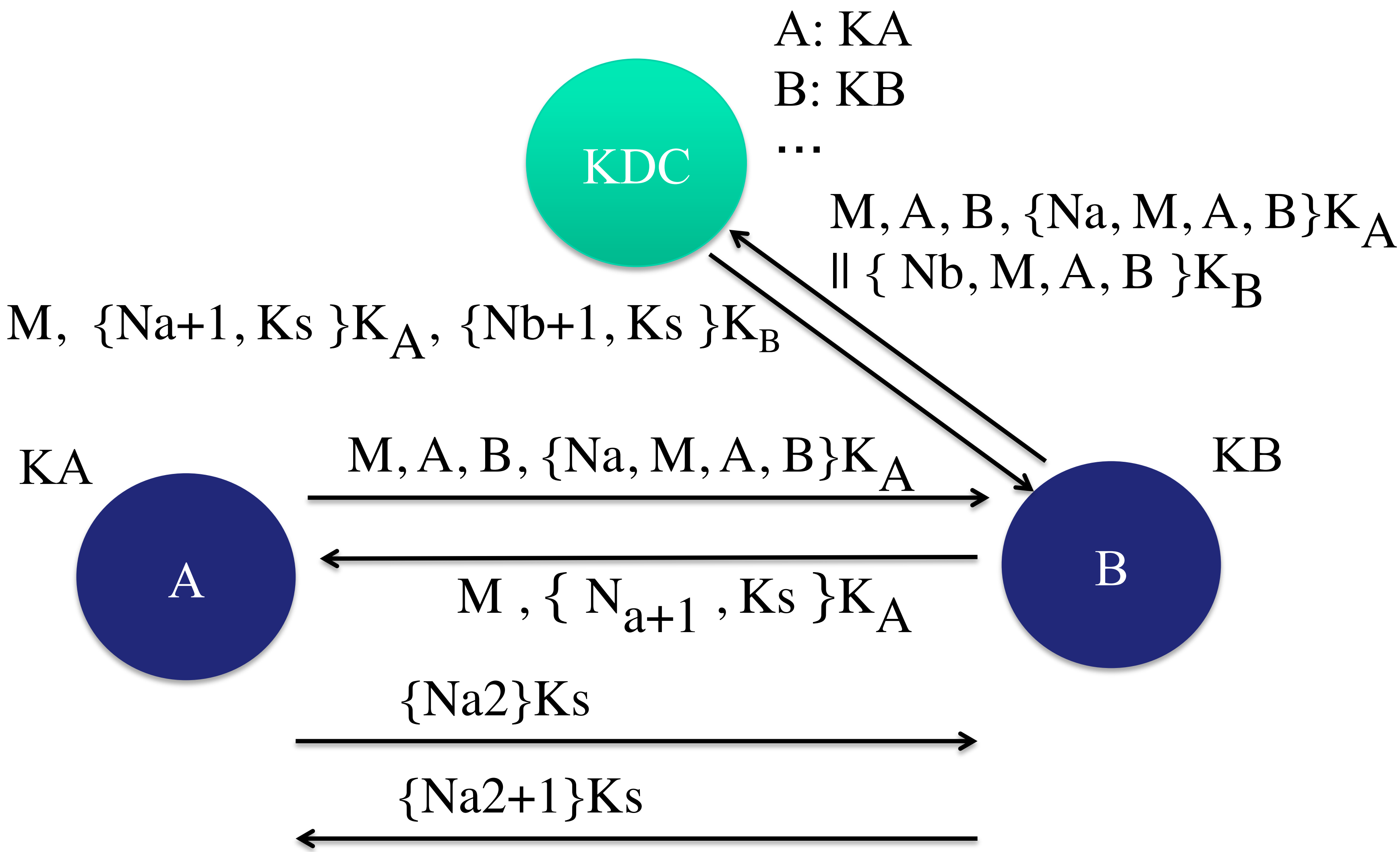
Fixed Needham-Schroeder Model



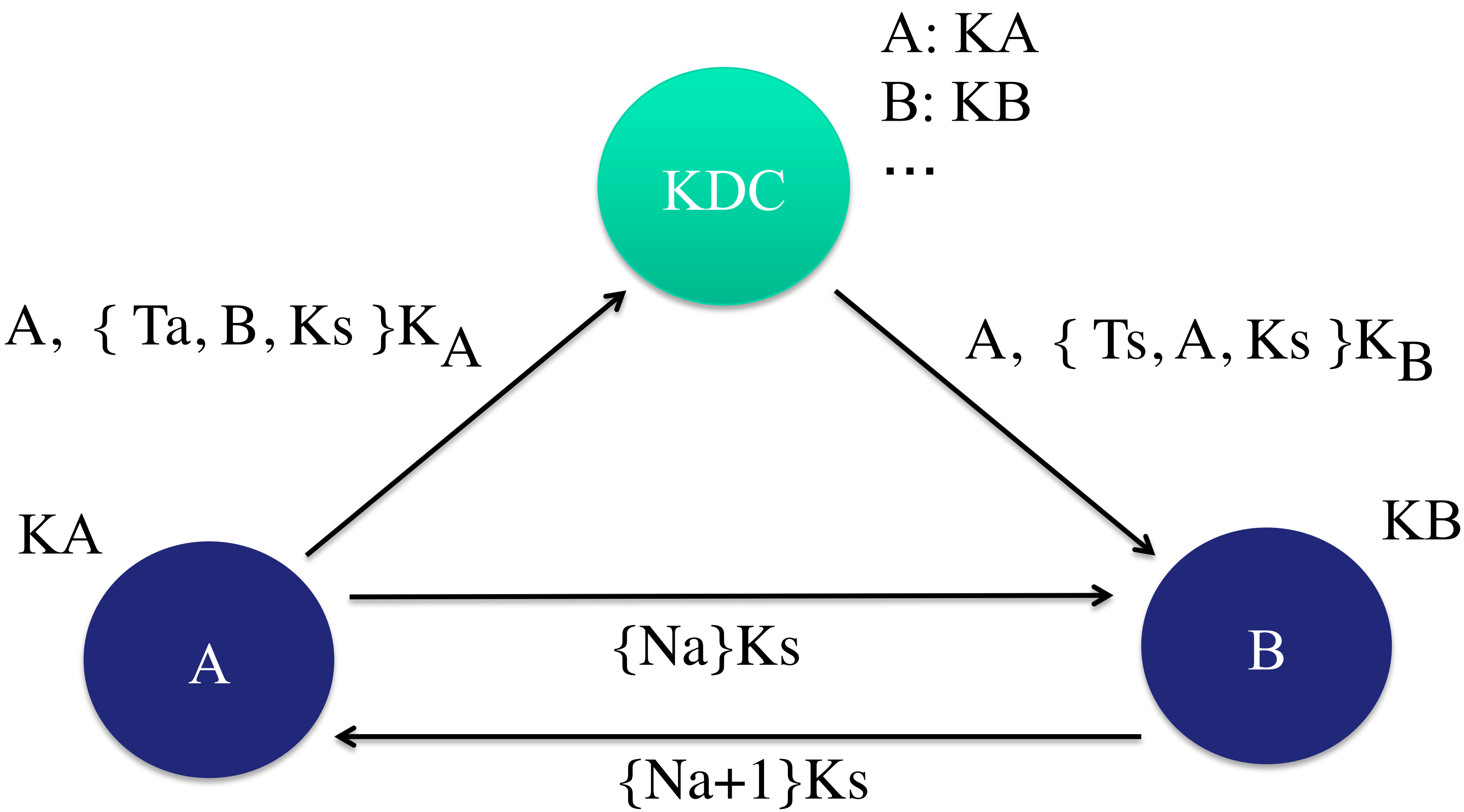
Alternative Needham-Schroeder Model



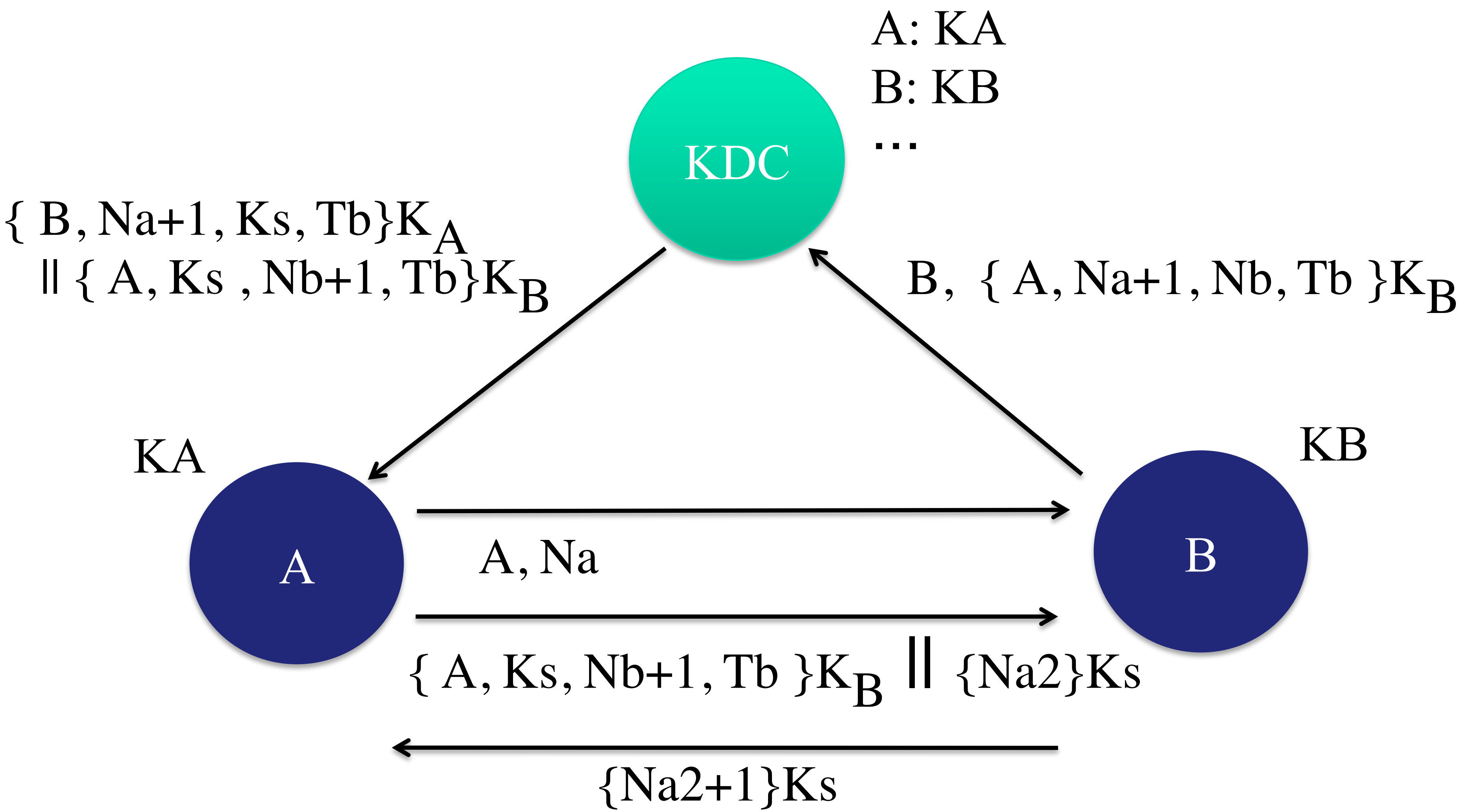
Otway-Rees Model



Wide-Mouth-Frog Model



Neuman-Stubblebine Model



Outline

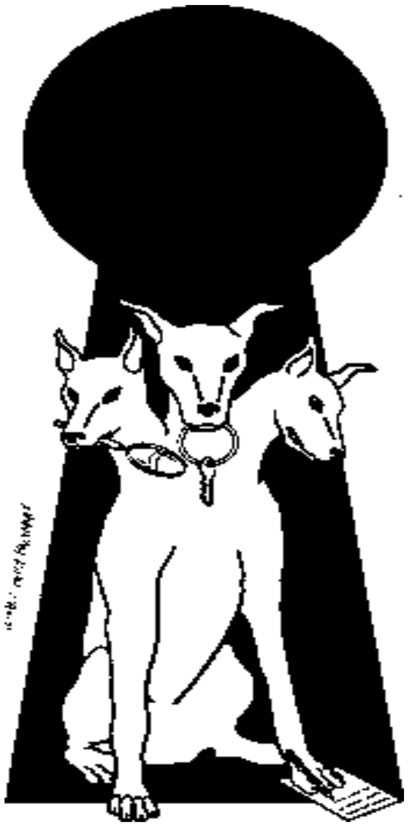
- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Kerberos

Kerberos Authentication and Key Distribution Protocol

Kerberos

In Greek mythology, a many headed dog, the guardian of the entrance of Hades



- MIT, Project Athena, Steve Miller and Clifford Neuman, Oct, 1988

  Dynamic (standardization) Evolution…

  - RFC 1510 Sep/1995

  ……

  - RFC 8009, Oct/2016

  …. (on going RFCs)

https://datatracker.ietf.org/doc/search/?name=Kerberos&activedrafts=on&rfcs=on

Kerberos Evolution

- Version 5, John Kohl and Neuman
  - RFC 1510 – 1993 (V4), made obsolete by RFC 4120, 2005 (V5)
- Until 2000, MIT implementations with DES banned from exportation by the US gov.
- KTH-KRB developed by the Royal Institute of Technology, Sweden, initially from the eBones MIT version (V4)
  - After RIT released V5 (Heimdal distribution)
- Kerberos implementations from MIT freely available after 2000
- Microsoft Windows 2000 adoption of Kerberos as default authentication protocol
- 2007, Kerberos Consortium (Sun, Apple, Google, Centrify, Microsoft, MIT, Stanford Univ and other founding sponsors)
- New kerberos improvements until now

What is Kerberos ?

- **Authentication service and Key Establishment Protocol**
  - Designed for use in a distributed environment
  - In fact an Authentication and Key Distribution Service dor Distributed Applications (C/S Model)
- **Following a SSO Authentication Approach for Client/Server applications**
  - Generic solution (SSO Single-Sign-On philosophy)
  - "Kerberized" applications
- **Separation of authentication concerns within the multiple entities involved:**
  - Clients, Servers
  - Servers (Kerberos Services)
  - Delegation between authentication domains (Kerberos Realms)

Outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Kerberos Requirements

First report and identified requirements as:

- **Security**
  - Protection against eavesdroppers trying to impersonate users and services
- **Reliability**
  - To avoid a single point of failures/attacks
  - Reinforced for a distributed architecture
- **Transparent**
  - Transparent for users (similar to non-kerberized client applications and local logon procedures)
    - Password-based authentication in the base line
- **Scalable**
  - Support for a large number of clients and servers, in a distributed environment
    - Modular architecture, supporting possible different administrative distributed domains

Security Concerns and Kerberos

Concerns:

- Session Key-Distribution with a generic SSO approach
  - **Authentication**, **confidentiality** and **timeliness**
    - **Message Replaying, Reflections, Re-Ordering, Flow-Control based attacks**

- Session key establishment for **confidentiality**:
  - **Authentication and session key establishment** protocol
    - Based on Symmetric Encryption Processes and KDCs
    - Requires the use of previously shared secret keys (master-keys)

Timeliness in Kerberos

Timeliness means:

- **Message "freshness" (for anti-replaying control)**
- Provided by
  - Using Sequence Numbers, Timestamps (with Secure and Trustable Sync. Time) and/or Challenge/Response Proofs (nonces)
    - Timestamps as controlled "nonces", no as "temporal-sync. clock assumptions)

KERBEROS (security concerns)

Users wish to access services on different servers, from workstations (LANs, Corporate Internetworked LANs, … or more generically, in a Internet Environment)

Possible threats:

- **Unilateral/Mutual authentication threats**
  - User pretend to be another user.
    - Fake-personification, Identity Spoofing
      - (userID authentication attack)
  - User alter the network address of a workstation (ex., IP spoofing, IP Masquerading)
  - Fake services pretend to be the "correct services" running in "correct machines" (masquerading of service names, service identifiers, IP spoofing)
- **Adversaries eavesdrop on message exchanges**
  - Confidentiality and integrity threats
- **Adversaries may trigger replay attacks.**
  - Message Replaying, to gain entrance or to disrupt operations

Outline

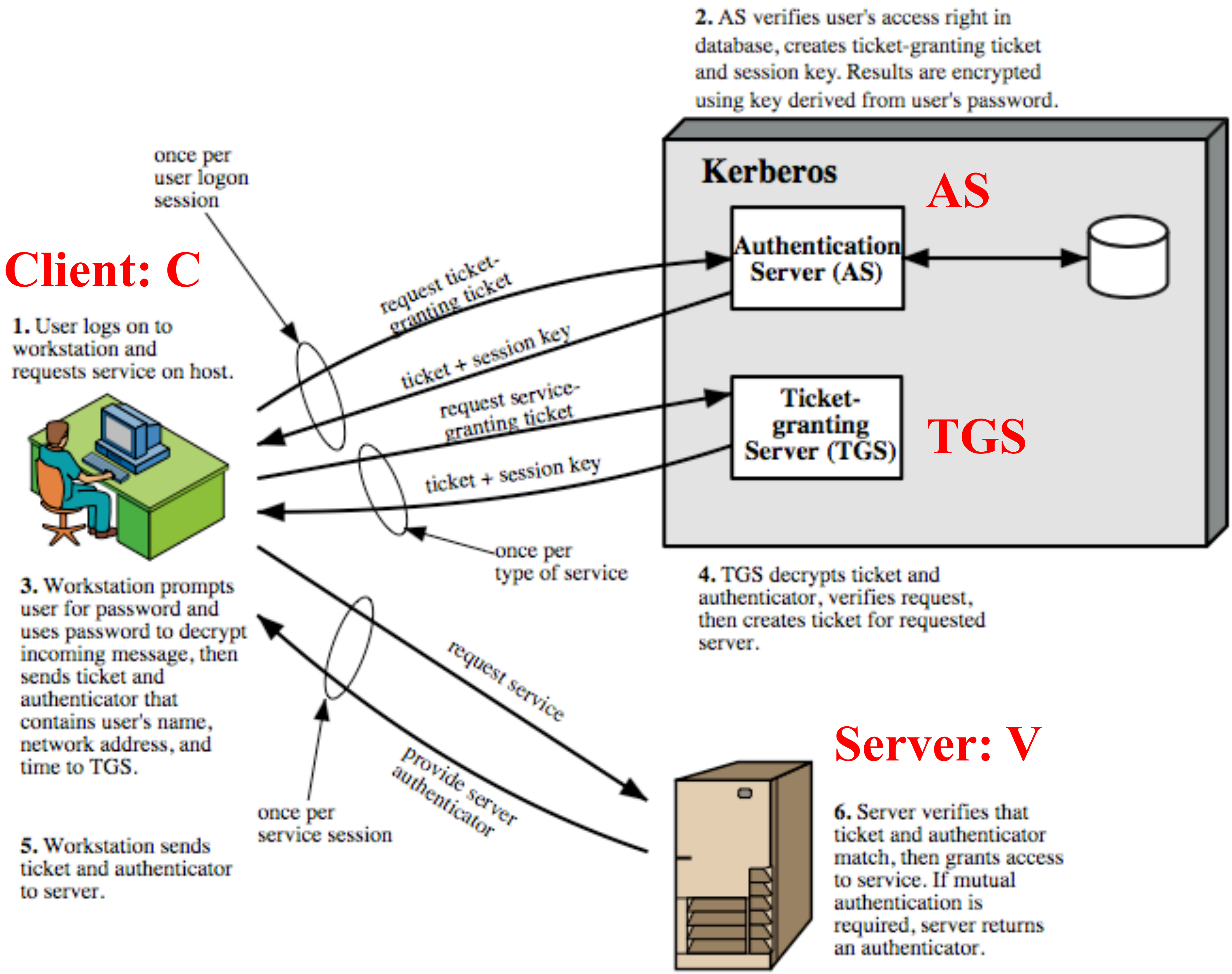
- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Kerberos base model entities

- Clients
- Servers

- Kerberos Service:
  - AS: Authentication Server
  - TGS: Ticket Granting Server

Obs) Can use one or more AS and one or more TGS


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
AS
Authentication Server (AS)
Ticket-granting Server (TGS)
TGS
Client: C
once per user logon session
1. User logs on to workstation and requests service on host.
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
request service
provide server authenticator
once per service session
Server: V
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos Dialogue and message exchanges

1. **Authentication Service Exchange**

   Obtain ticket granting ticket from AS

   - Once per session (once per user authenticated logon session)

2. **Ticket-Granting Service Exchange**

   Obtain service granting ticket from TGT

   - For each distinct service required
   - Once per type of service

3. **Client/server authentication exchange**

   To obtain service

   - On every service request
   - Once per specific service session

Outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Kerberos Version 4

Terms:

- C = Client
- AS = authentication server
- V = server
- ID_c = identifier of user on C
- ID_v = identifier of V
- P_c = password of user on C
- $T(P_c)$ = transformation of a verifiable password or secret with protection: ex., OTP, or $\{P_c\}_{kc\text{-}as}$
- AD_c = network address of C
- K_v = secret encryption key shared by AS an V
- TS = timestamp
- || = concatenation

Kerberos V4 Protocol

(1) C → AS	$ID_c \parallel ID_{tgs} \parallel TS_1$
(2) AS → C	$E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
	$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS	$ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
(4) TGS → C	$E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$
	$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$
	$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
	$Authenticator_c = E(K_{c,tgs}, [ID_C \parallel AD_C \parallel TS_3])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V	$Ticket_v \parallel Authenticator_c$
(6) V → C	$E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)
	$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
	$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$

© I

(c) Client/Server Authentication Exchange to obtain service

V4 Shortcomings

- **Encryption system dependence (V4, only DES)**
  - Inclusion of Encryption Type Identifier
  - Encryption Keys tagged with type and length
- **Internet protocol dependence (V4, only IP)**
  - ISO network addresses, tagged with type and length
- **Message byte ordering, message representation types (V4: specific tags, specific implementation types)**
  - Lack of standardization for generic adoption (ASN.1 and BER)
- **Ticket lifetime and control (granularity issues)**
  - V4, 8 bits as units of 5 minutes
- **Authentication forwarding or delegation (no support)**
  - Forwarding client credentials from server to server, and other flexibility/adaptive issues
- **Scalable inter-domain authentication (no support)**

More Kerberos V4 limitations

- **Double Encryption (tickets encrypted twice)**
  - Messages 2 and 4, Second encryption not necessary
- **PCBC encryption mode**
  - Propagating Cipher Block Chaining
  - Not standard and vulnerable (security)
  - V5 uses CBC
- **Session keys**
  - Replaying messages from old sessions to the client and to the server
  - No rekeying possibility specified for each client/server connection
- **Password Based Attacks**

Outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

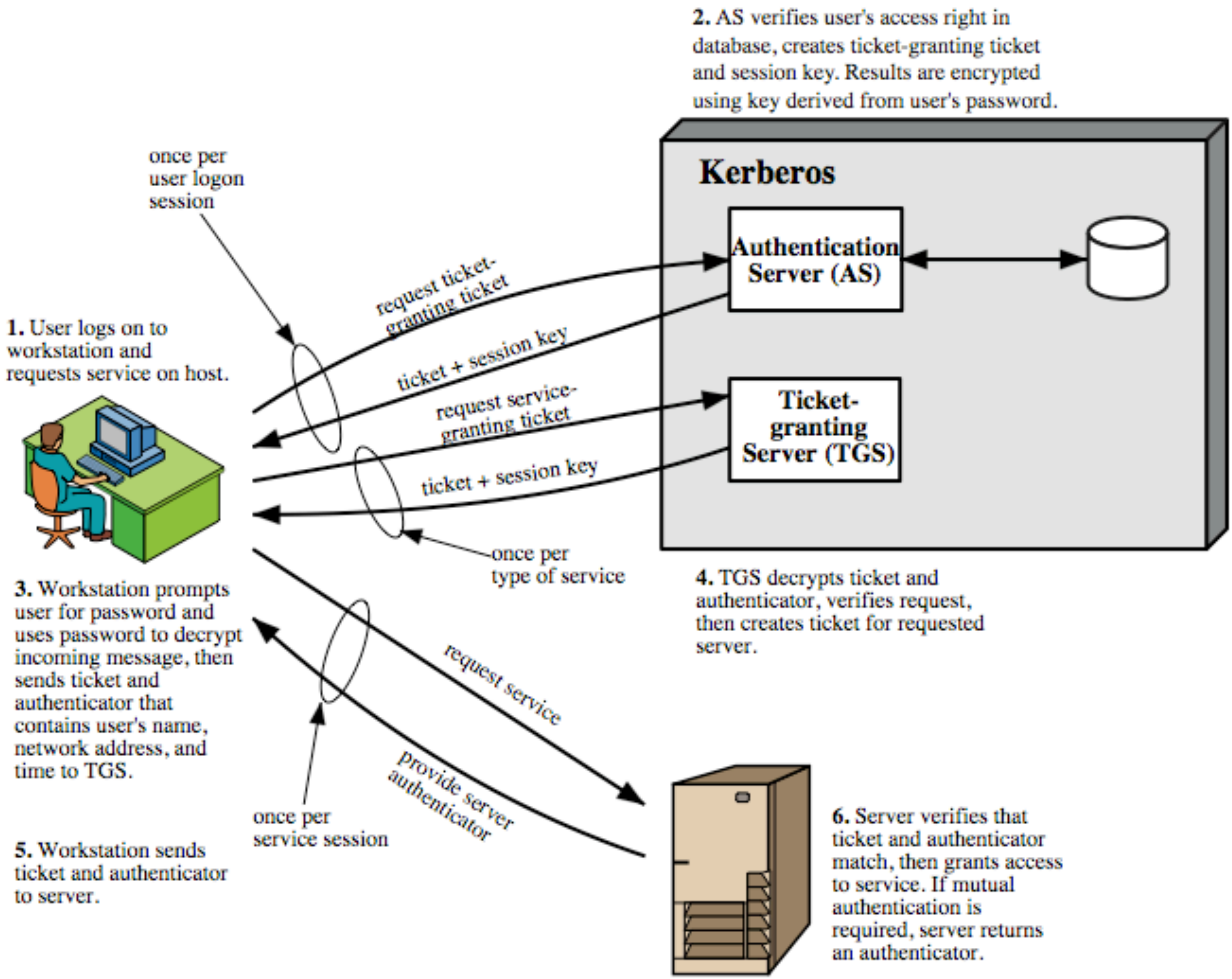
Kerberos Version 5

- Developed in mid 1990's to overcome V4 shortcomings and drawbacks
- Specified as Internet standard RFC 1510
- Improvements:
  - Addresses environmental shortcomings
    - Encryption algorithm, network protocol, byte order, ticket lifetime, authentication forwarding and inter-realm authentication
  - And some technical deficiencies
    - Double encryption, non-std mode of use, session keys, password attacks

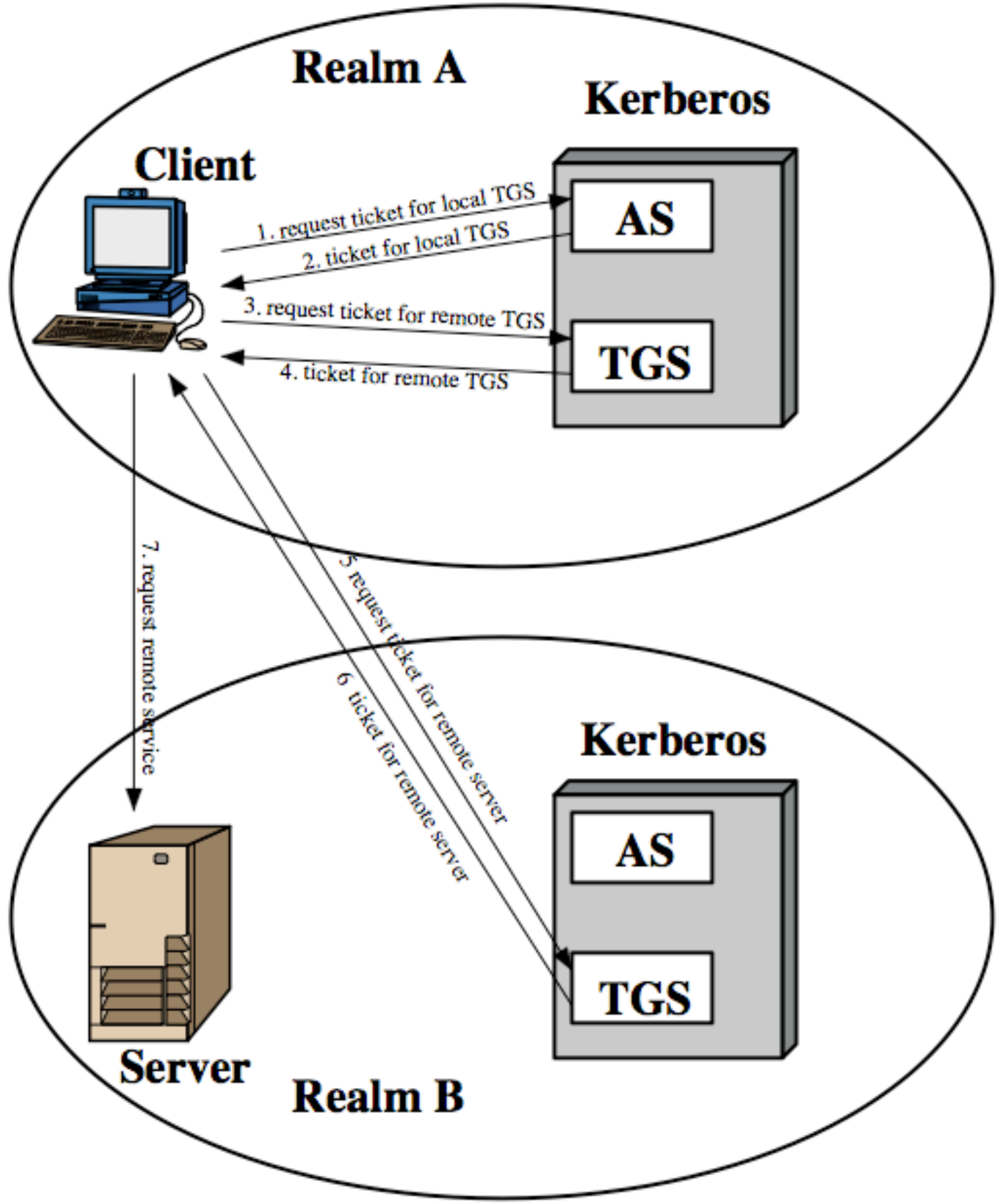
Kerberos Realms

- A Kerberos V5 REALM environment consists of:
  - A Kerberos server (AS + TGS)
  - A number of clients, all registered with server
  - Application servers, sharing keys with server
- A realm is typically a single administrative domain

- If have multiple realms, their Kerberos servers must share keys and trust each other
  - TGS in one realm issues TGS tickets to remote TGS in another realm
  - Implicit delegation model
    - AS authenticated clients in one realm
    - TGS tickets issued for other TGS (in other realm)


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
once per user logon session
Authentication Server (AS)
request ticket-granting ticket
ticket + session key
1. User logs on to workstation and requests service on host.
request service-granting ticket
Ticket-granting Server (TGS)
ticket + session key
once per type of service
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Request for Service in Another Realm



© DI/FCT/UNL, (updated for 2º Sem, 19/20

Kerberos protocol (version 5)

(1) **C → AS** Options ∥ ID_c ∥ $Realm_c$ ∥ ID_{tgs} ∥ $Times$ ∥ $Nonce_1$

(2) **AS → C** $Realm_c$ ∥ ID_C ∥ $Ticket_{tgs}$ ∥ $E(K_c, [K_{c,tgs} \parallel Times \parallel Nonce_1 \parallel Realm_{tgs} \parallel ID_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [Flags \parallel K_{c,tgs} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) **C → TGS** Options ∥ ID_v ∥ $Times$ ∥ ∥ $Nonce_2$ ∥ $Ticket_{tgs}$ ∥ $Authenticator_c$

(4) **TGS → C** $Realm_c$ ∥ ID_C ∥ $Ticket_v$ ∥ $E(K_{c,tgs}, [K_{c,v} \parallel Times \parallel Nonce_2 \parallel Realm_v \parallel ID_v])$

$$Ticket_{tgs} = E(K_{tgs}, [Flags \parallel K_{c,tgs} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times])$$

$$\mathrm{Ticket}_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{Flags} \parallel \mathrm{K}_{c,v} \parallel \mathrm{Realm}_c \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{Times}])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \parallel Realm_c \parallel TS_1])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) **C → V** Options ∥ Ticket_v ∥ $\mathrm{Authenticator}_c$

(6) **V → C** $\mathrm{E}_{K_{c,v}}$ [TS_2 ∥ Subkey ∥ Seq#]

$$\mathrm{Ticket}_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{Flags} \parallel \mathrm{K}_{c,v} \parallel \mathrm{Realm}_c \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{Times}])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \parallel Realm_c \parallel TS_2 \parallel Subkey \parallel Seq\#])$$

(c) Client/Server Authentication Exchange to obtain service

©

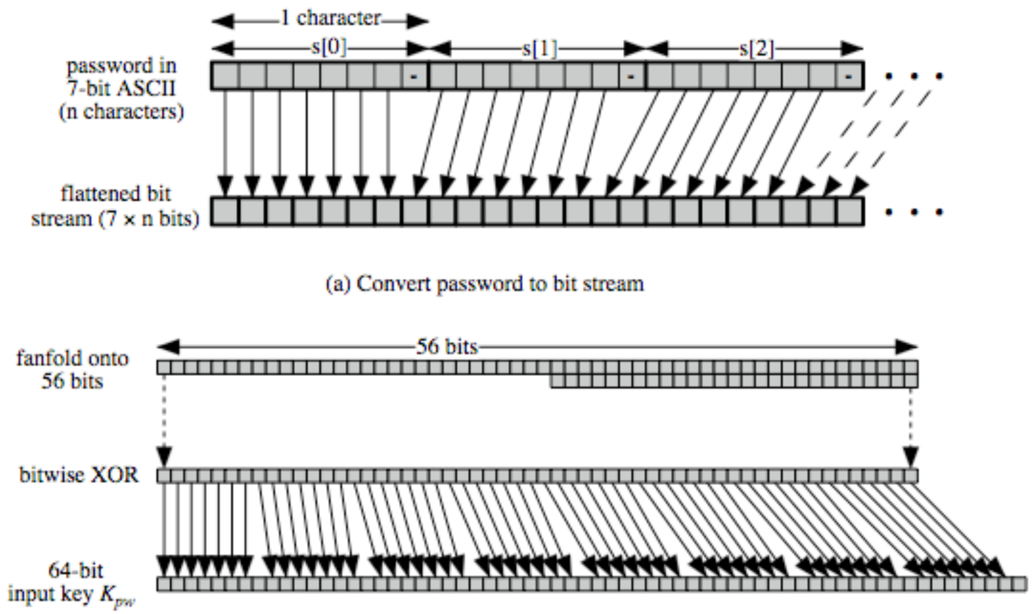
Kerberos V5 flags

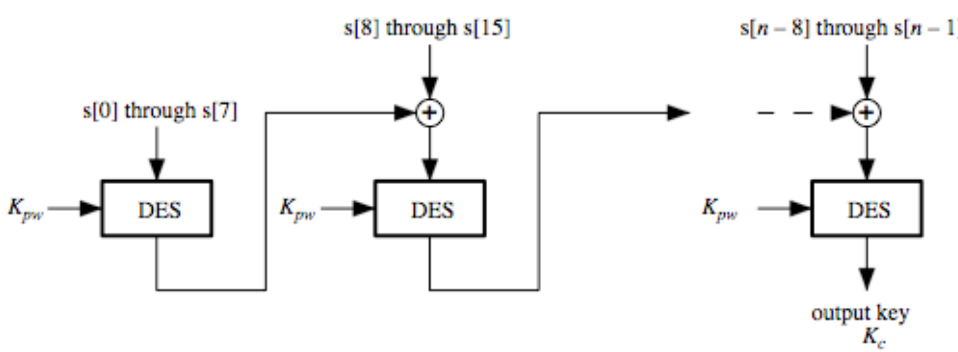
INITIAL	This ticket was issued using the AS protocol and not issued based on a ticket-granting ticket.
PRE-AUTHENT	During initial authentication, the client was authenticated by the KDC before a ticket was issued.
HW-AUTHENT	The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client.
RENEWABLE	Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date.
MAY-POSTDATE	Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket.
POSTDATED	Indicates that this ticket has been postdated; the end server can check the authtime field to see when the original authentication occurred.
INVALID	This ticket is invalid and must be validated by the KDC before use.
PROXIABLE	Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket.
PROXY	Indicates that this ticket is a proxy.
FORWARDABLE	Tells TGS that a new ticket-granting ticket with a different network address may be issued based on this ticket-granting ticket.
FORWARDED	Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket.

© DI/FC

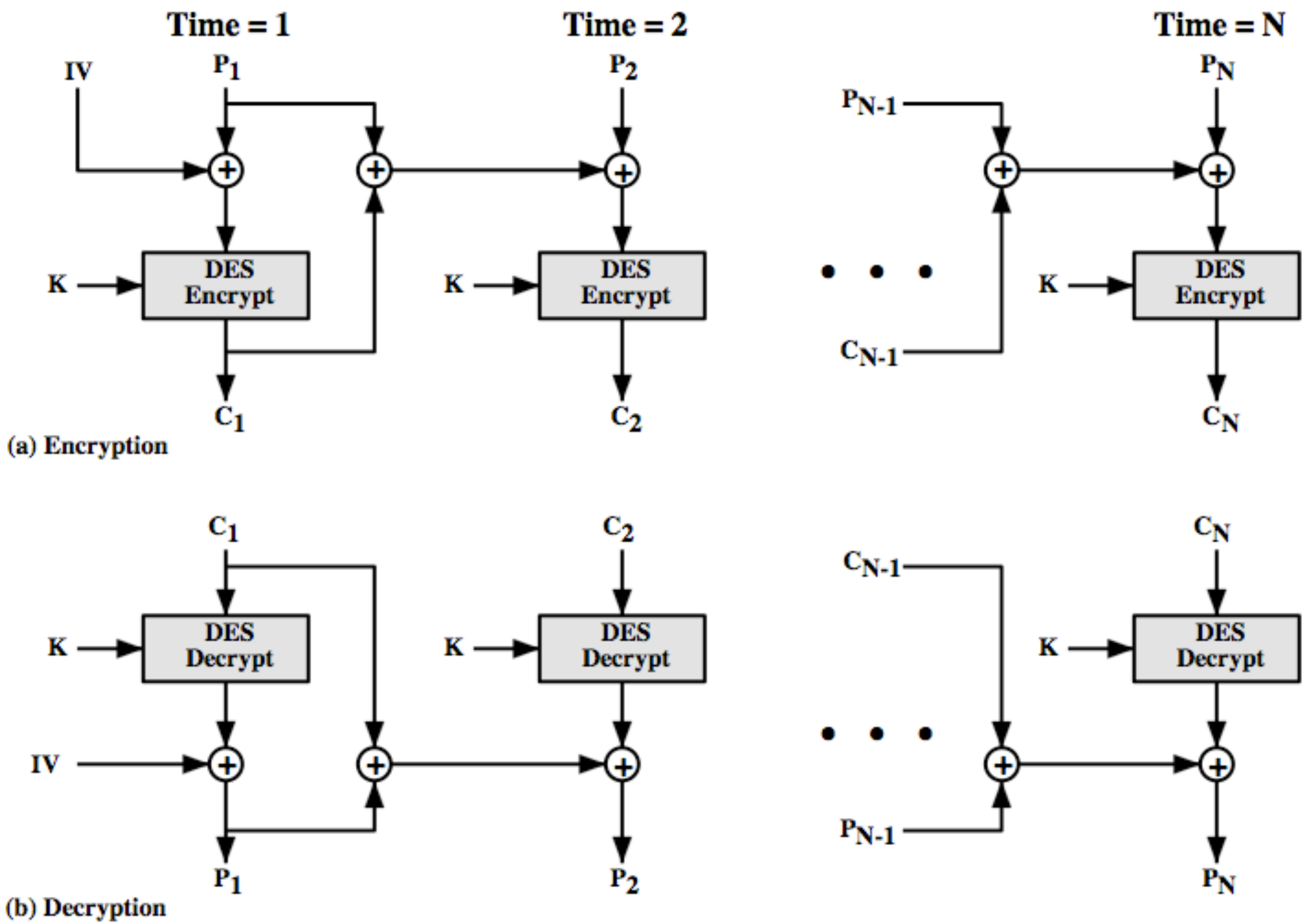
Kerberos Authentication and Encryption Techniques: can be weak … why ?

DES-CBC Checksum of Encryption Scheme to generate Encryption Key from the Password





PCBC Mode initially adopted



Kerberos Summary and Use

Two Kerberos versions:

- 4 : restricted to a single realm
- 5 : allows inter-realm authentication
- Kerberos v5 is an Internet standard
- specified in RFC1510, and used by many utilities
- Some defined variants (ex., PKINIT Keberos)

Use of Kerberos:

- Based on a KDC solution (divided in AS and TGS)
- Need to have Kerberised applications running on all participating systems
- Major problem - US export restrictions, Password-Attacks (Key-Generation process)
- Kerberos cannot be directly distributed outside the US in source format (& binary versions must obscure crypto routine entry points and have no encryption)
- Crypto libraries reimplemented locally

Outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Key Distribution Protocols and Models
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Improvements

- Microsoft additions (RFC 3244)
  - MS Win2000 Kerberos Change Password and Set Password Protocols
- Microsoft RFC 4757, adoption of RC4
- Encryption and checksum option schemes: RFC 3961
- AES in Kerberos V5: RFC 3962
- Kerberos V5 More detail in message definition and specification: RFC 4120 (RFC 1510 is now obsolete)
- Kerberos V5 with GSS-API: RFC 4121

- Public Key Authentication for the Message Authentication Exchange
  - PKINIT Based Kerberos RFC 4556

PKINIT Variants

- PK-INIT
  - Kerberos Initial Ticket Acquisition using Public Key
    - Certificates or Raw Key Pairs
    - See A. Jaggard, A. Scedrov, Jor-Kay Tsay, Computationally Sound Mechanized Proof of PKINIT for Kerberos,

http://dimacs.rutgers.edu/~adj/Research/papers/jst07fcc.pdf
- PKI Integration proposals
- PK-CROSS
  - Establishment of Kerberos Cross Realm relationships using Public Key
    - Mutual Authentication of TGSs
    - Secure Generation of Static Keys
- PK-APP (aka KX509)*
  - Acquisition of Public Key certificates using Kerberos

Covered outline

- **Key (or security association parameters) distribution using symmetric encryption**
  - Initial solutions for the Key-Distribution problem
  - Solutions with a KDC (Key Distribution Center)
  - Key Management Issues
  - Kerberos Protocol for Authentication and Key Establishment
    - System Model and Overview
    - Kerberos Entities
    - Kerberos Protocol Version 4
    - Kerberos Protocol Version 5
    - Kerberos variants and improvements

Revision: Suggested Readings

Suggested Readings:

W. Stallings, Network Security Essentials – Applications and Standards, Chap 4., sections 4.1, 4.2 and 4.3

Optional References for Kerberos

Optional / Other References:

www.whatis.com (search for kerberos)
Bryant, W. Designing an Authentication System: A Dialogue in Four Scenes. http://web.mit.edu/kerberos/www/dialogue.html
Kohl, J.; Neuman, B. "The Evolotion of the Kerberos Authentication Service" http://web.mit.edu/kerberos/www/papers.html
http://www.isi.edu/gost/info/kerberos/