

Authentication Scenarios

- Users share a password with a trusted authority (authentication servers)

Kerberos

Challenge-
response
Symmetric
Key

Kerberos

- Kerberos provides authentication in distributed systems
 - Guarantees safe access to network resources (e.g., printer, databases etc.)
 - There is a central authority that allows to reduce the number of passwords that users must memorize
- Reference:
 - proposed by MIT
<http://web.mit.edu/kerberos/www/dialogue.html>
 - free download in US and Canada (after 2000, in most locations)
 - widespread use (most operating systems)

Kerberos

- Scenario: A needs to access service provided by B
 - Authentication of A
 - Optional: authentication of B
 - Optional: decide session keys for secret communication and/or authentication
- C is trusted server (authority that shares keys with A and B)
- Idea: use ticket to access services; tickets are valid in each time window



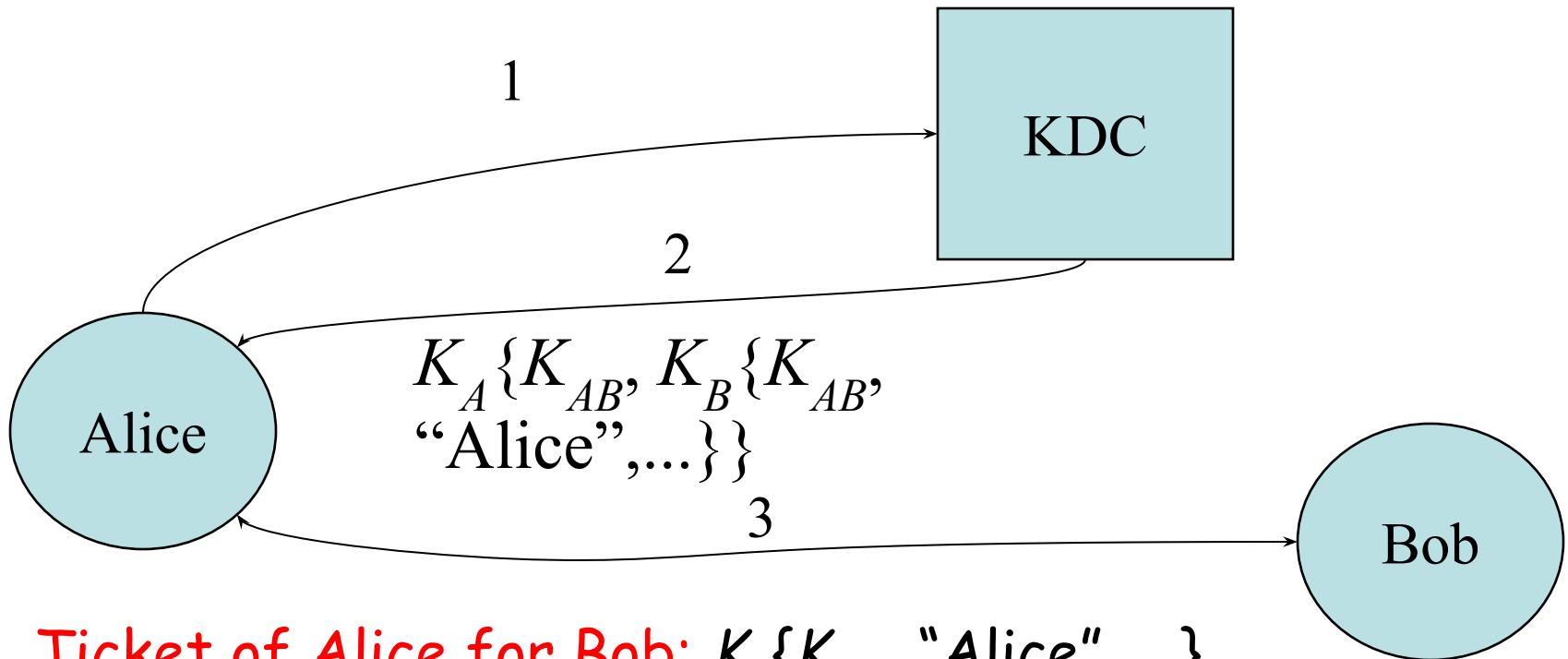
Kerberos

- KDC (Key Distribution Center) is the server (both trusted and physically safe)
- Messages are safe with respect to cryptographic attacks and data integrity
- Kerberos provides security for applications like
 - telnet
 - rtools (rlogin, rcp, rsh)
 - network file systems (NFS/AFS)
 - e-mail
 - printer servers
 - ...

Preliminaries

- Each user (also named as principal) has a master secret key with KDC
 - for human users master secret key is derived from password
 - for system resources, keys are defined while configuring the application
- Each principal is registered by the KDC
- All master keys are stored in the KDC database, *encrypted with the KDC master key*

Tickets, Alice, Bob, KDC



Ticket of Alice for Bob: $K_B \{ K_{AB}, \text{"Alice"}, \dots \}$,
 K_A master key of Alice, K_B master key of Bob,
 K_{AB} session key to be used by A and B
only Bob can decrypt and checks the message

Tickets

- a ticket is encrypted with the secret key associated with the service
- ticket basically contains
 - sessionkey
 - username
 - client network address
 - servicename
 - lifetime
 - timestamp

Kerberos: simplified version

9

A asks for a ticket TicketB for B

1. A sends to C: A, B, N (N nonce)
2. C sends to A: TicketB, $K_{AC}(K_{AB}, N, L, B)$
L = "lifetime of ticket"
3. [A checks N and knows ticket lifetime]
A sends to B: TicketB, $K_{AB}(A, t_A)$ [authenticator]
4. [B checks that A's identity in TicketB and in authenticator are the same, time validity of ticket]
5. B sends to A: $K_{AB}(t_A)$
[in this way shows knowledge of t_A]

TicketB = $K_{BC}(K_{AB}, A, \text{"lifetime"}, \text{timestamp}), N \text{ nonce}, K_{AB}$
session key; "lifetime" = validity of ticket; t_A timestamp

Session key and Ticket-gr anting Ticket (TGT)

- Messages between host and KDC should be protected using the master key (**derived from user's password**)
 - For each request to the KDC:
 - **user must type the password each time**, or
 - **user's password is temporarily stored** (to avoid the user the need of retyping)
- all above solutions are **inadequate!**

Session key and Ticket-granting Ticket (TGT)

Proposed solution to reduce # of times user types the password and/or master key

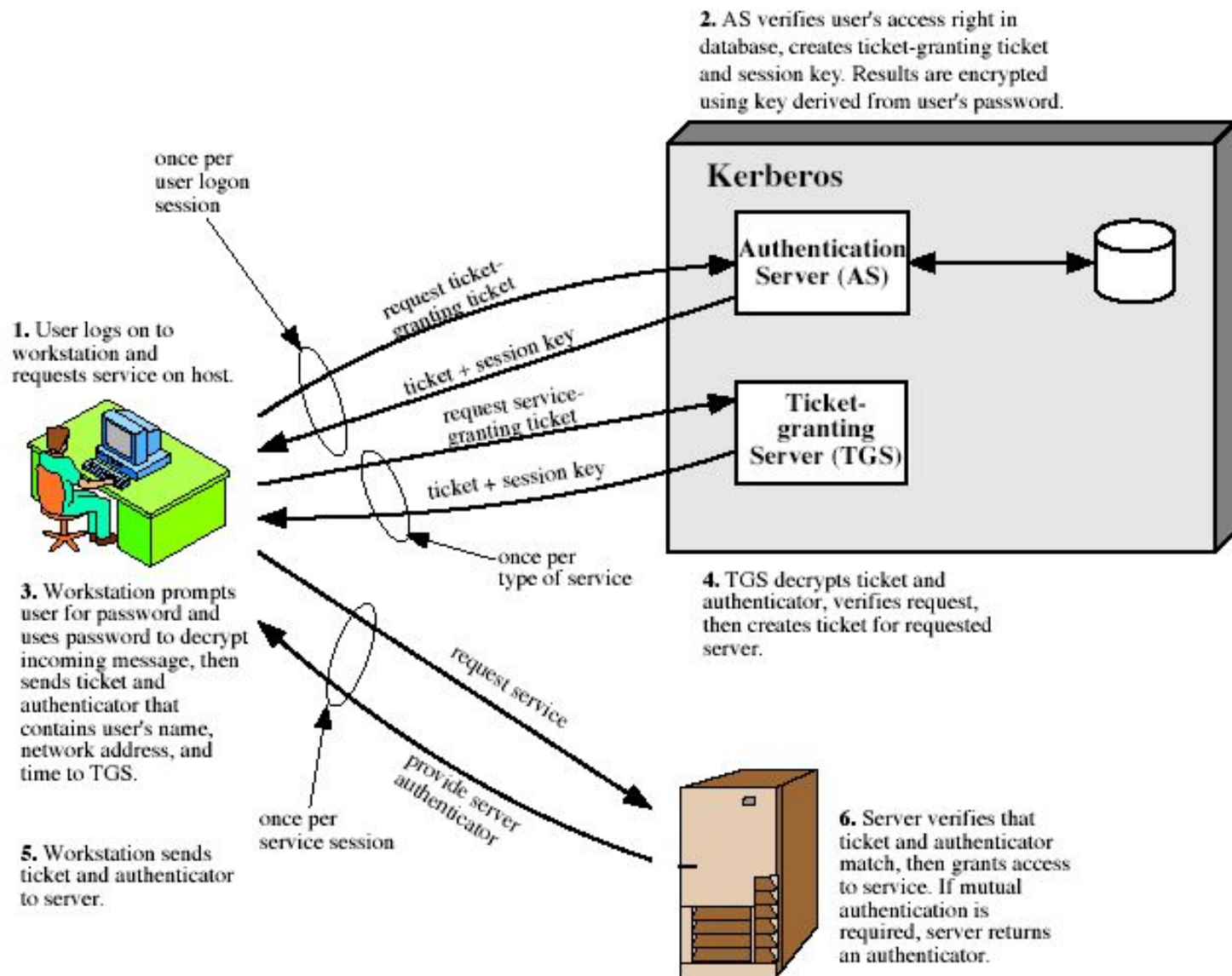
at initial login a session key S_A is derived for Alice by KDC

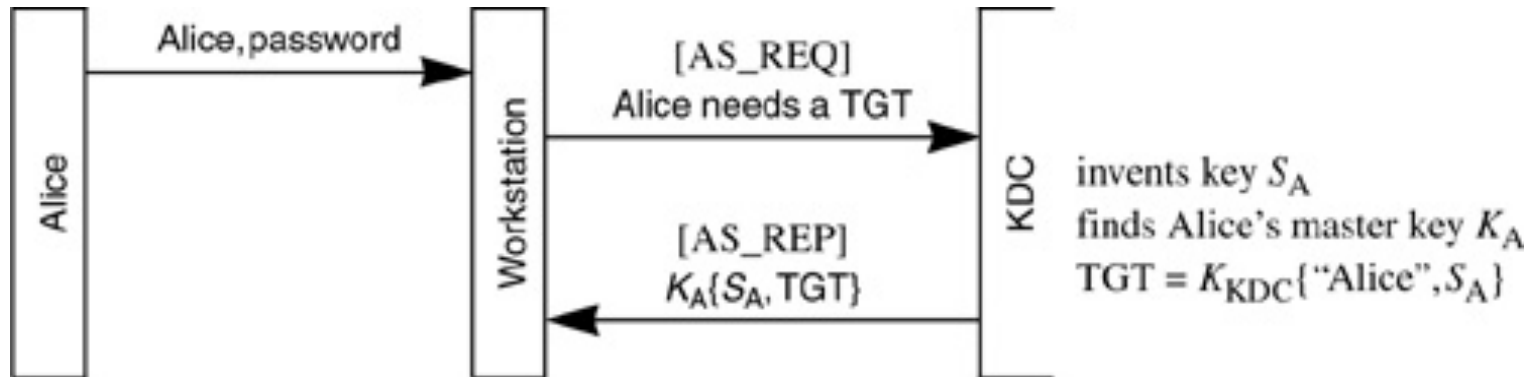
S_A has a fixed lifetime (e.g., 1 day, 4 hours)

KDC gives Alice a TGT that includes session key S_A and other useful information to identify Alice (encrypted with KDC's master key)

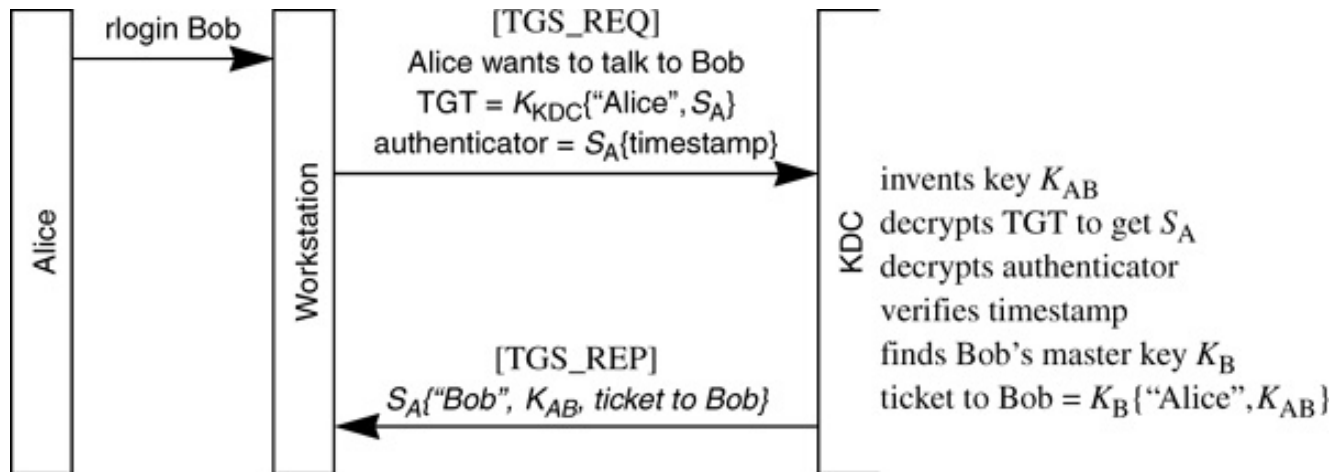
Session key and Ticket-granting Ticket (TGT)

- Subsequent requests from Alice to KDC use TGT in the initial message
- Subsequent tickets provided by KDC for accessing server V are decrypted using K_{VC}
- User provides password only once
- No password is stored

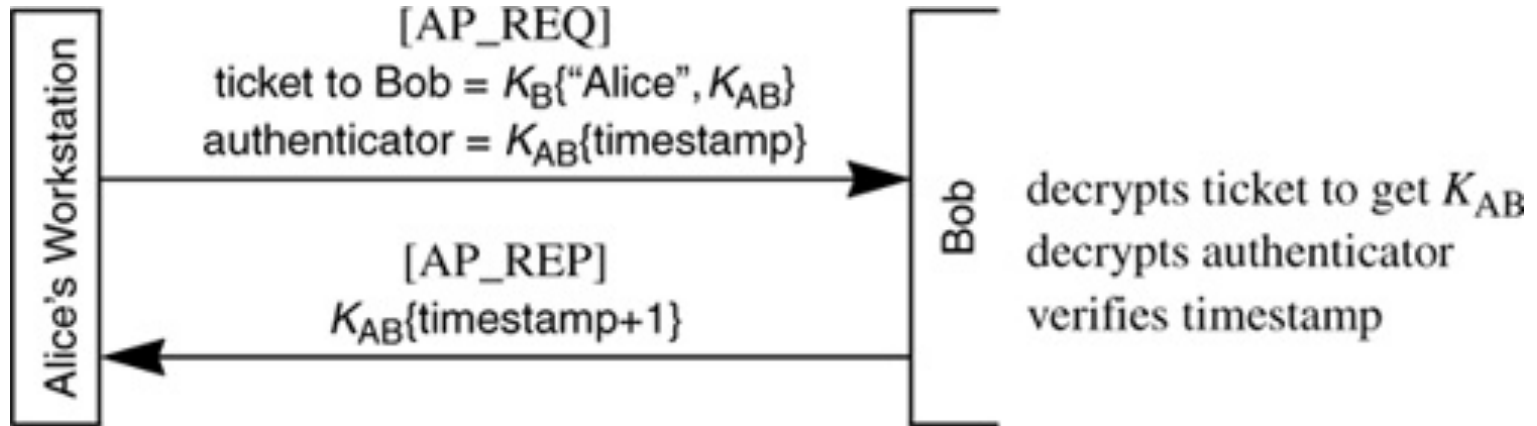




Login



Ticket request



Use of ticket

Authentication and time synchronization

- Authenticator: $K_x\{\text{timestamp}\}$
 - K_x is a session key
- Global Synchronous Clock is required
- Authenticator is used to avoid
 - replay of old messages sent to the same server by the adversary (old messages are eliminated)
 - replay to a server (when there are many servers)
 - Authenticator DOES NOT guarantee data integrity (a MAC is required)
- Vulnerability: many instances of same server all using same master key. Replay attack!
 - how could it be avoided?

KDC and TGS

- KDC and TGS are similar (the same?) why do we need two different entity?
 - Historical reasons
 - One KDC can serve different systems (1 KDC many TGS)
- multiple copies of KDC, sharing same KDC master key - availability and performance
- Consistency issues in KDC databases
 - A single KDC stores information concerning principal (safer)
 - Periodically upload information to other KDC

Kerberos - Performance

- KDC stores only TGT and tickets
- Most work is on client
- KDC is involved only at login to provide TGT
- KDC uses only permanent information

Message types

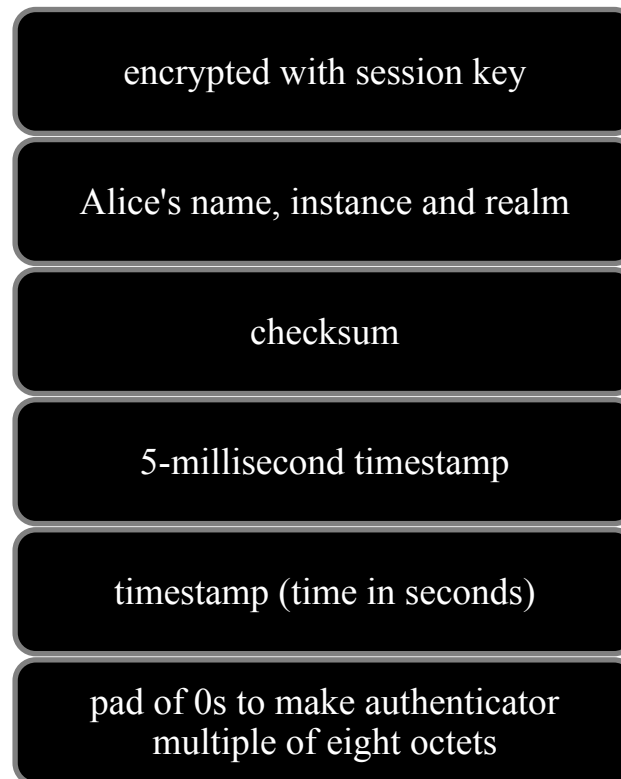
- AS_REQ
 - Used when asking for the initial TGT.
- AS_REPLY (also TGS_REP)
 - Used to return a ticket, either a TGT or a ticket to some other principal.
- AP_REQ (also TGS_REQ)
 - Used to talk to another principal (or the TGS) using a ticket (or a TGT).
- AP_REQ_MUTUAL
 - This was intended to be used to talk to another principal and request mutual authentication. In fact, it is never used; instead, applications know whether mutual authentication is expected.
- AS_ERR
 - Used for the KDC to report why it can't return a ticket or TGT in response to AS_REQ or TGS_REQ.
- PRIV
 - This is a message that carries encrypted integrity-protected application data.
- SAFE This
 - is a message that carries integrity-protected application data.
- AP_ERR
 - Used by an application to report why authentication failed.

Ticket (Alice, Bob)

encrypted with Bob's key

- Alice's name, instance and realm
- Alice's Network Layer address
- session key for Alice, Bob
- ticket lifetime, units of 5 minutes
- KDC's timestamp when ticket made
- Bob's name and instance
- pad of 0s to make ticket length multiple of eight octets

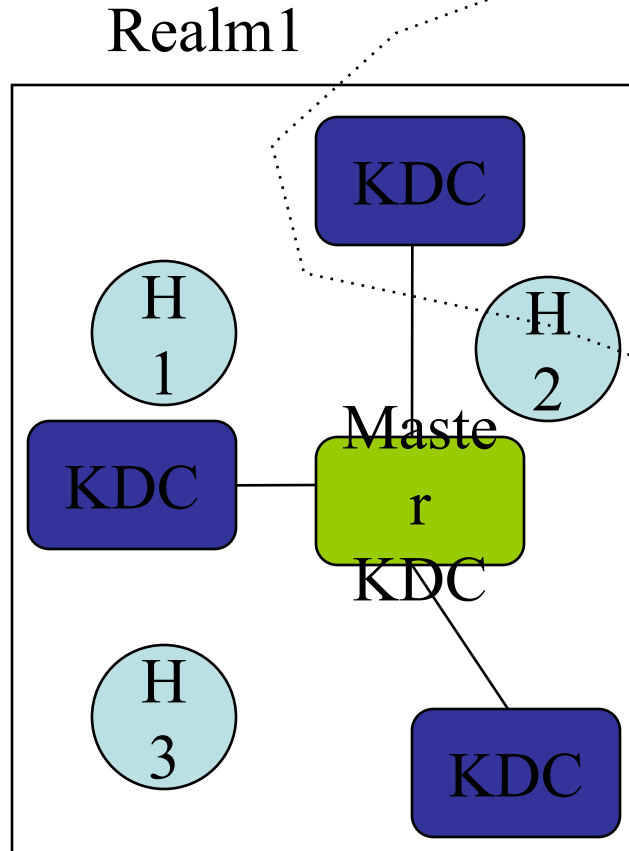
Authenticator



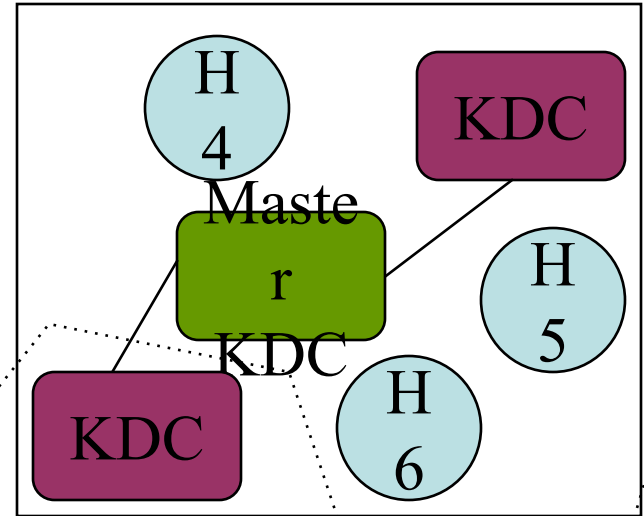
Kerberos: Realms

- In very large systems security and performance issues suggest to use not only a domain but more (several many KDC)
- REALM
- each realm has a different master KDC
- all KDCs share the same KDC master key
- two KDCs in different realms have different databases of users

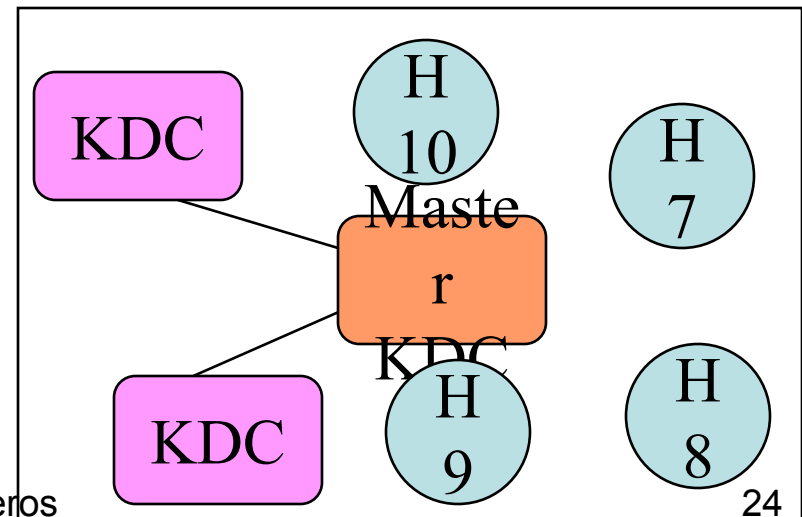
Kerberos V. 4: Realms



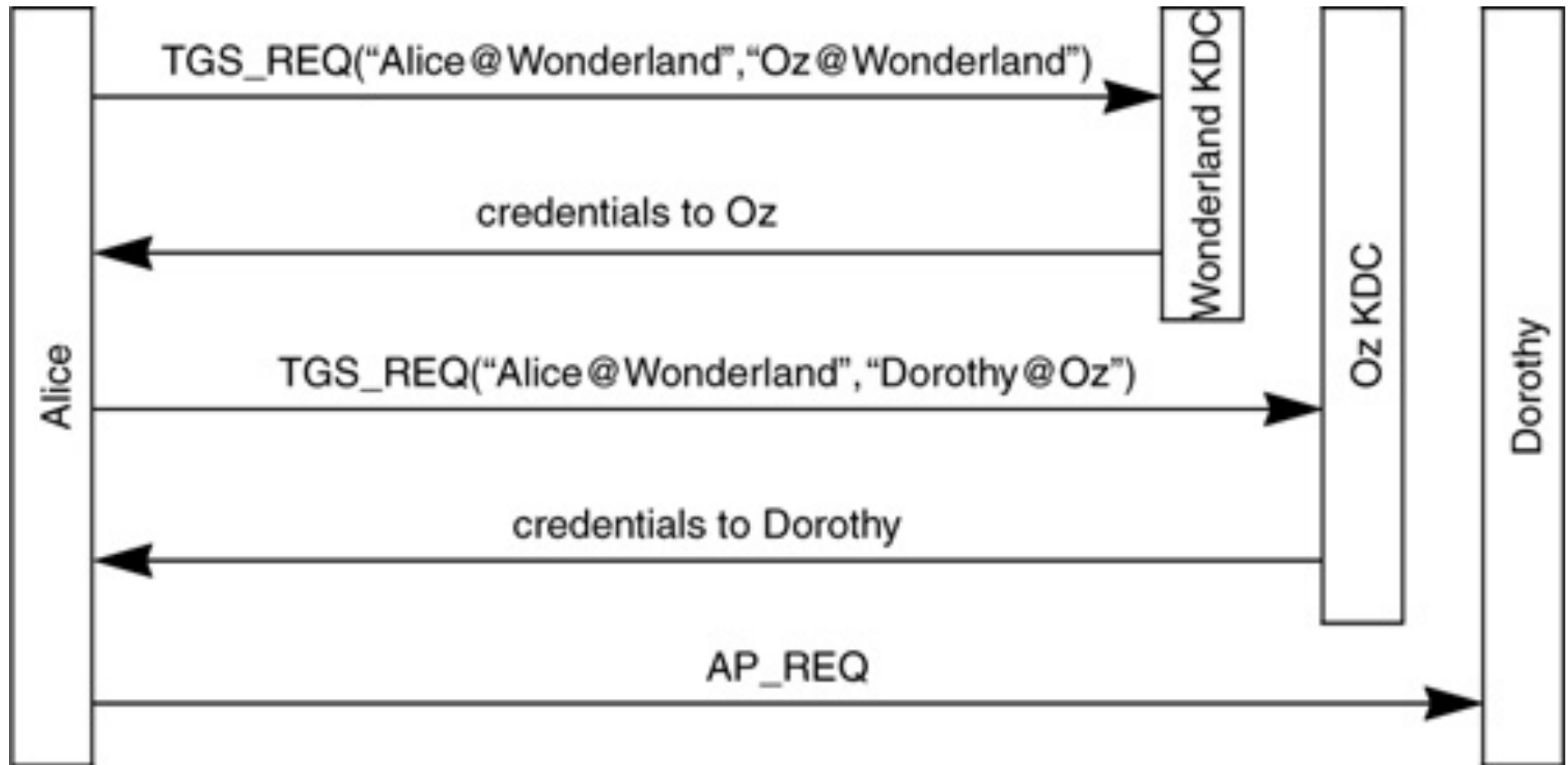
Realm2



Realm3



Authentication between realms



Other features

key version numbers

-

encryption for privacy and integrity

-

encryption for integrity only

Kerberos: version 5

- Same philosophy
- Major changes
- Integrity of messages, authentication using nonce (not only timestamps)
- Flexible encoding: many optional fields,
 - allows future extensions
 - overhead
- Major extensions to the functionality
- Delegation of rights: Alice allows Bob to access:
 - her resources for a specified amount of time
 - a specific subset of her resources
- Renewable tickets: tickets can be used for long time
- More encryption methods (Kerberos designed for DES)
- Hierarchy of realms

