Authentication Scenarios

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

Kerberos

Challenge-response Symmetric Key

- What do we learn from previous attacks?
- Timestamps: are valid only in a small time window
- Sequence numbers attached to messages are useful (to avoid replication attacks)
- Nonce: we should carefully use them (and we require good random number generator)

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 a given time window

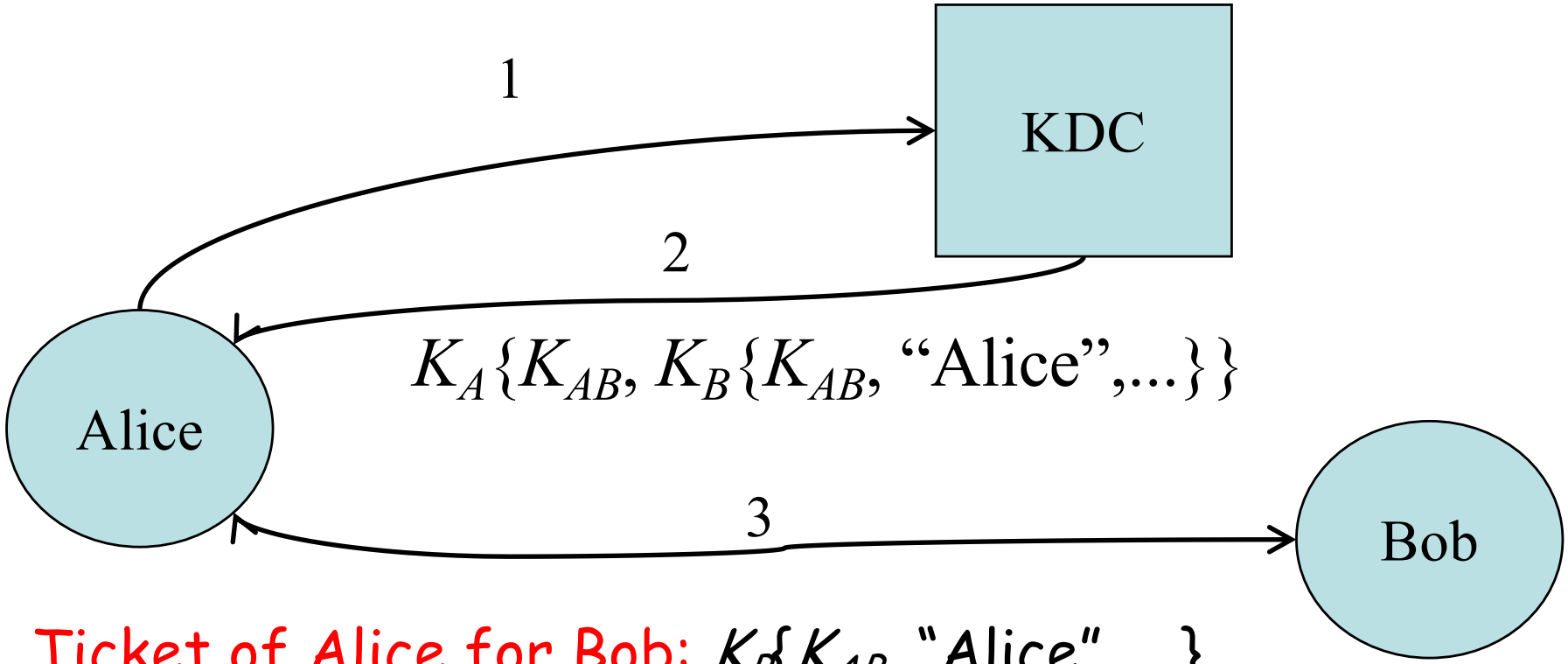
Kerberos

- 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: $K_B\{K_{AB}, \text{“Alice”}, \ldots\}$,
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 is able to decode 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

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, "lifetime", timestamp), N
nonce, K_{AB} session key; "lifetime"= validity of ticket; t_A timestamp

Session key and Ticket-granting 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!

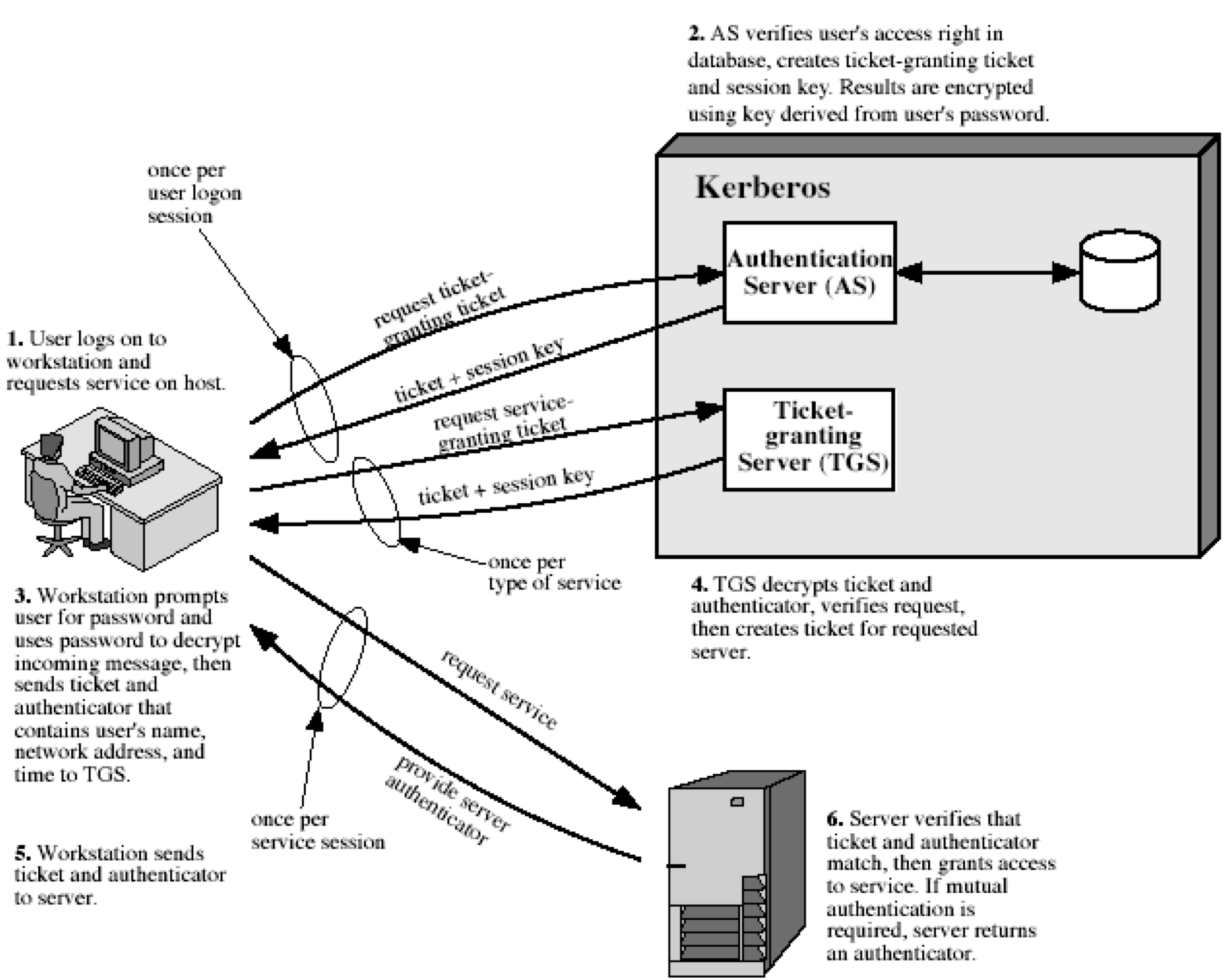
Session key and Ticket-granting Ticket (TGT)

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

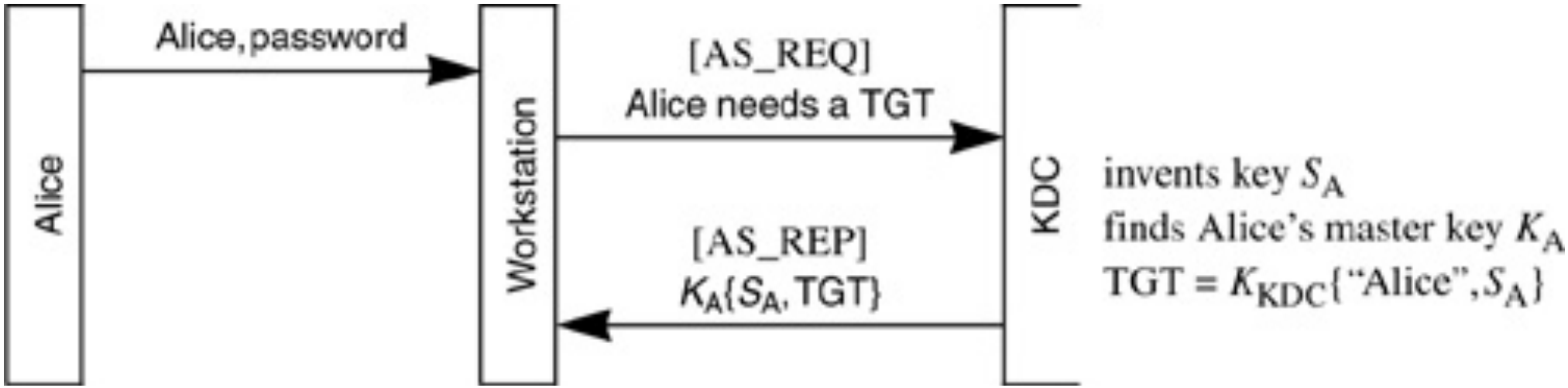
- at initial login a session key S_A is derived for Alice by KDC
- S_A has a fixed lifetime (e.g., 1 day, 4 hours)
- KDC gives Alice a TGT that includes session key S_A and other useful information to identify Alice (encrypted with KDC's master key)

Session key and Ticket-granting Ticket (TGT)

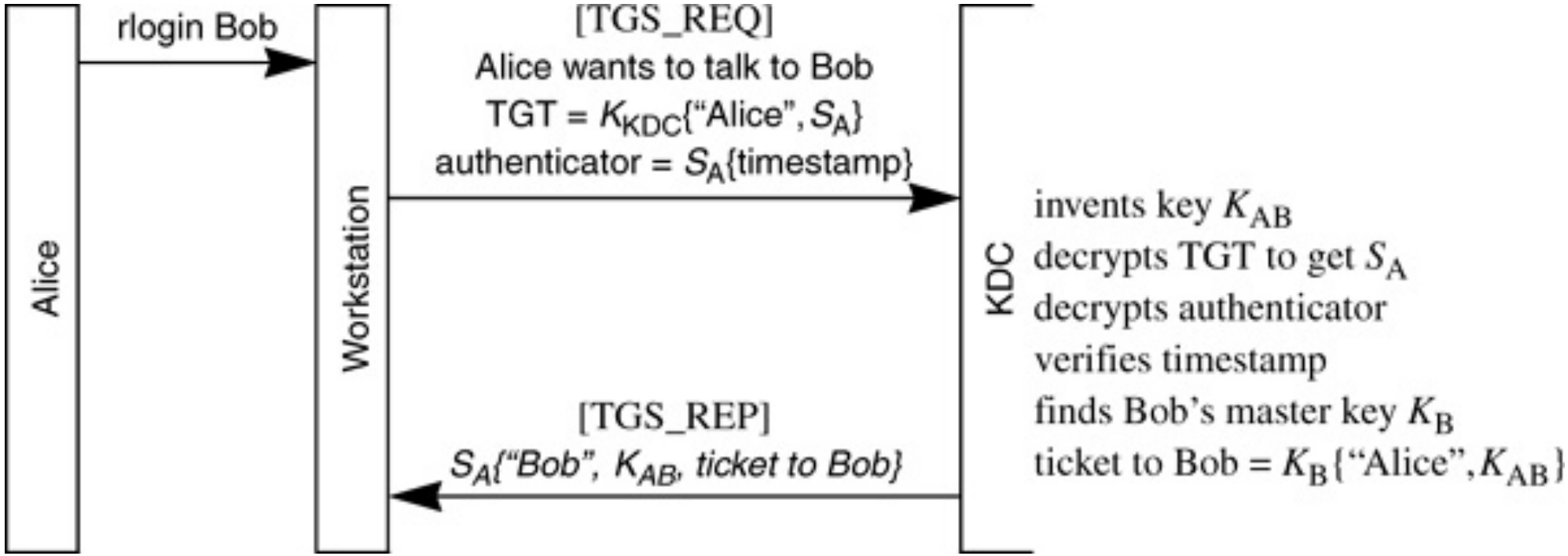
- Subsequent requests from Alice to KDC use TGT in the initial message
- Subsequent tickets provided by KDC for accessing server V are decoded using K_{VC}
- User provides password only once
- No password is stored


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
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
1. User logs on to workstation and requests service on host.
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.

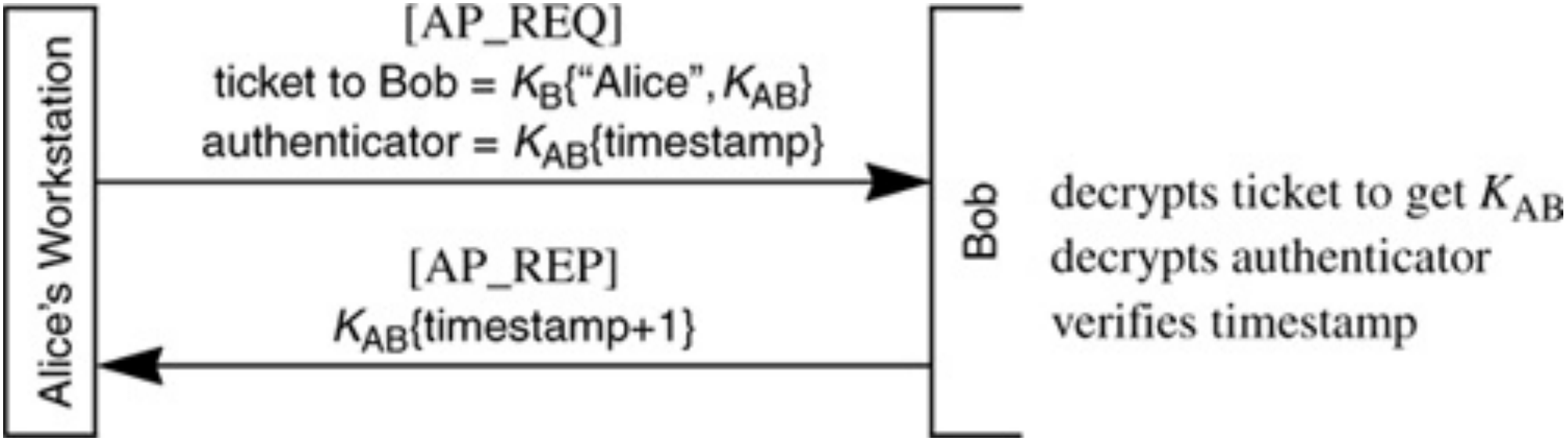
Login


Alice
Alice, password
Workstation
[AS_REQ]
Alice needs a TGT
KDC
invents key S_A
finds Alice's master key K_A
TGT = K_{KDC}{"Alice", S_A}
[AS_REP]
$K_A\{S_A, TGT\}$

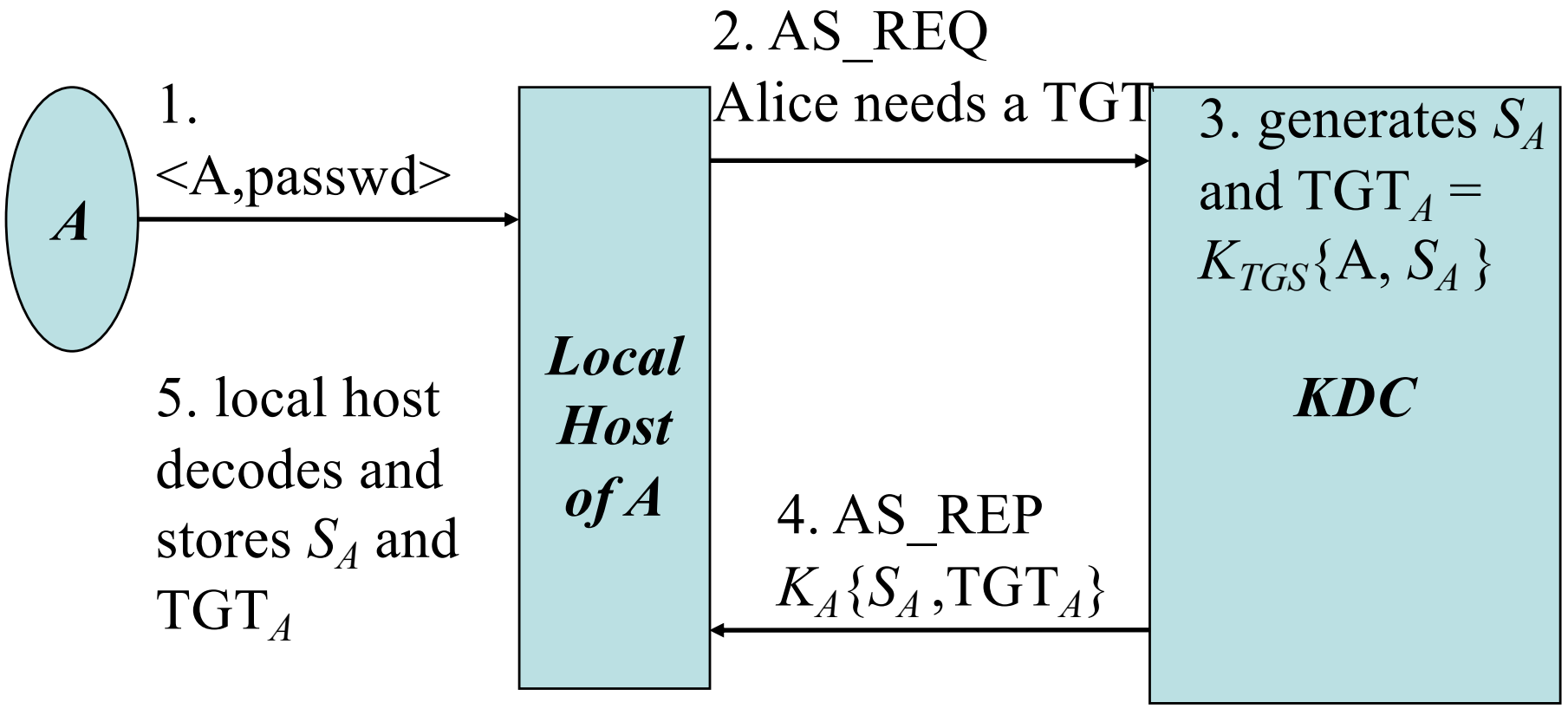
Ticket request



Use of ticket

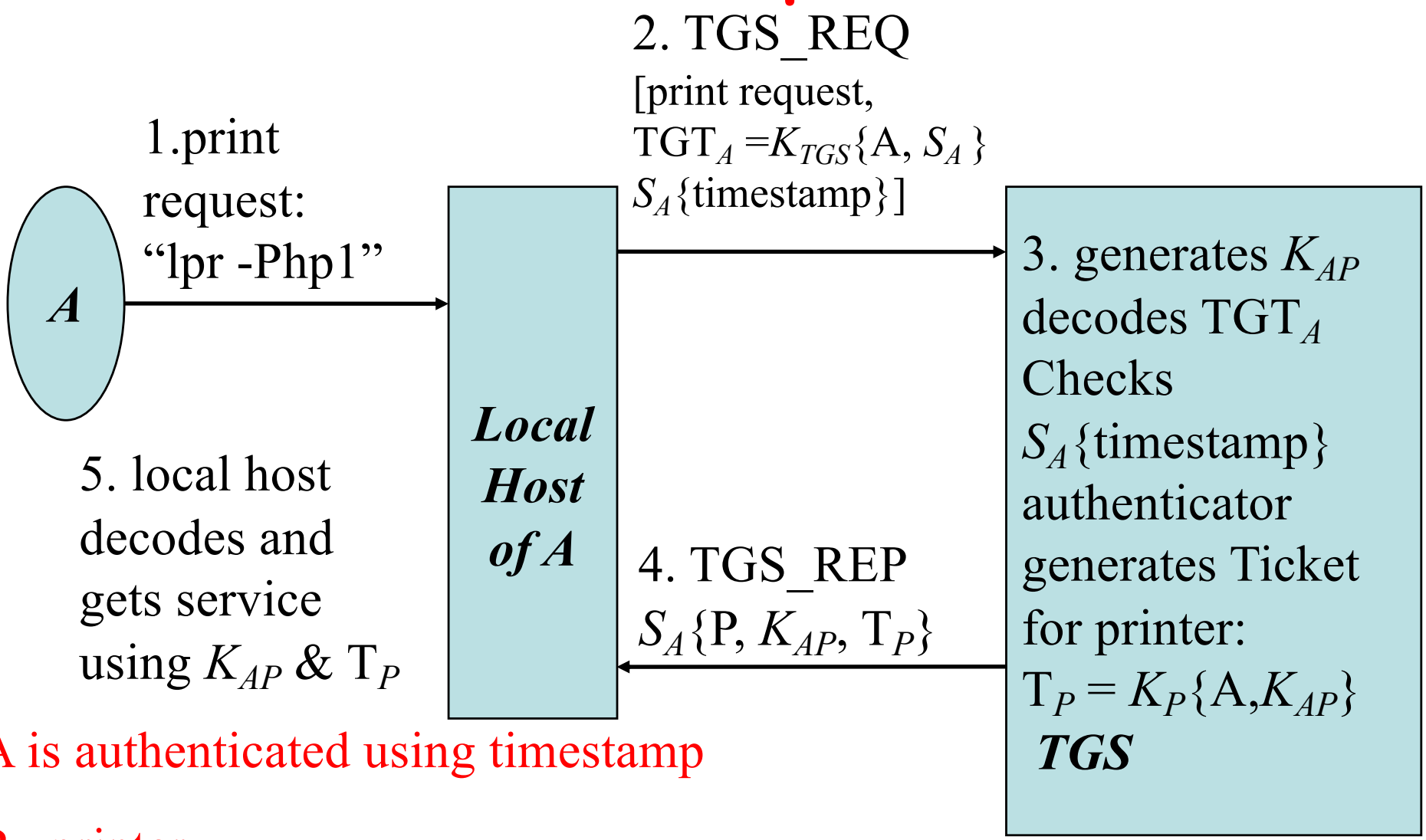


Login



- *Local host of A= (current) Alice's workstation*
- S_A = session key for A

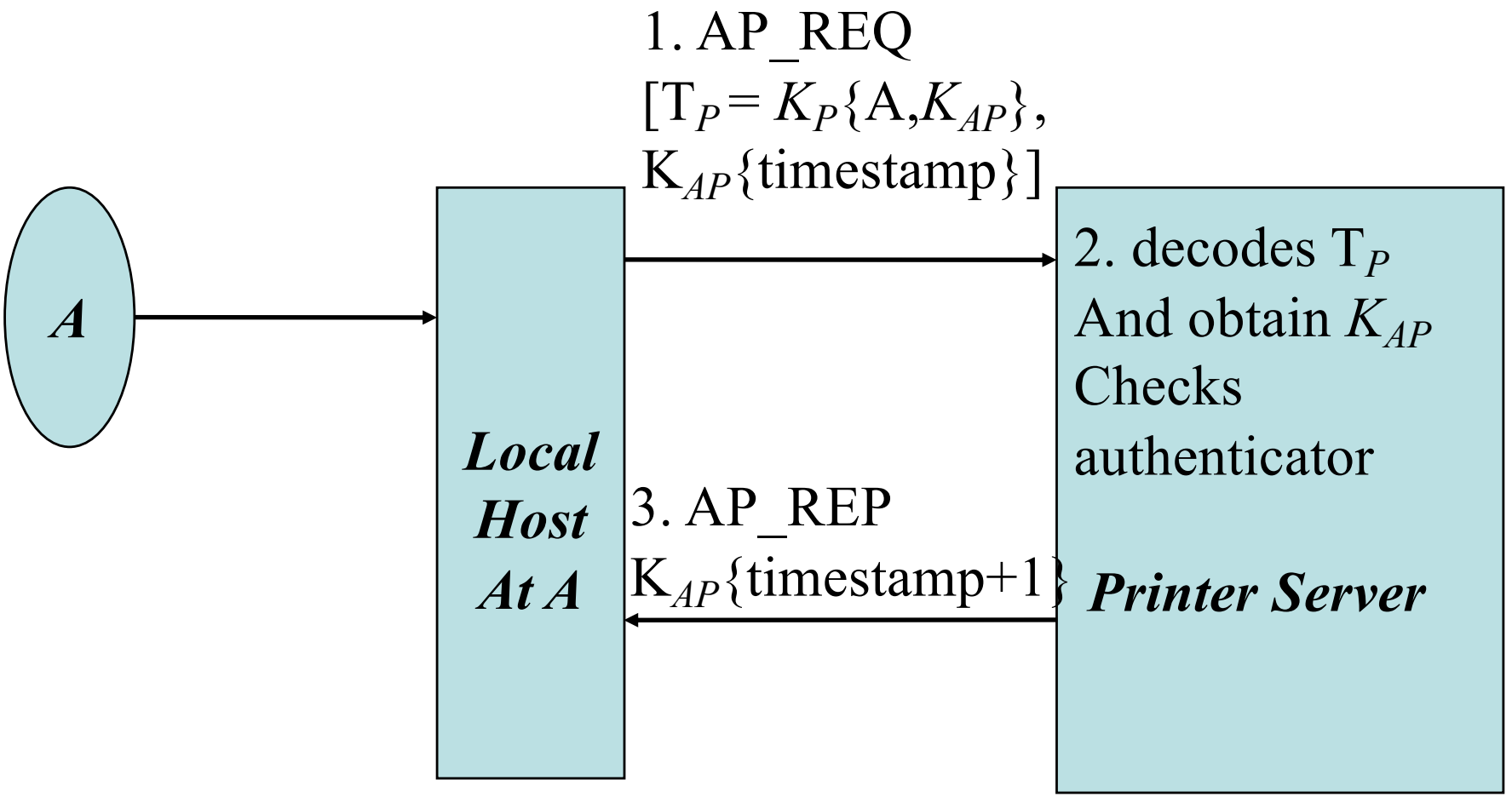
Ticket request



A is authenticated using timestamp

P =printer

Use of Ticket for printer P



- printer request is managed by A's local host
- there is mutual authentication using timestamp

Authentication and time synchronization

- Authenticator: $K_X\{timestamp\}$
  - 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

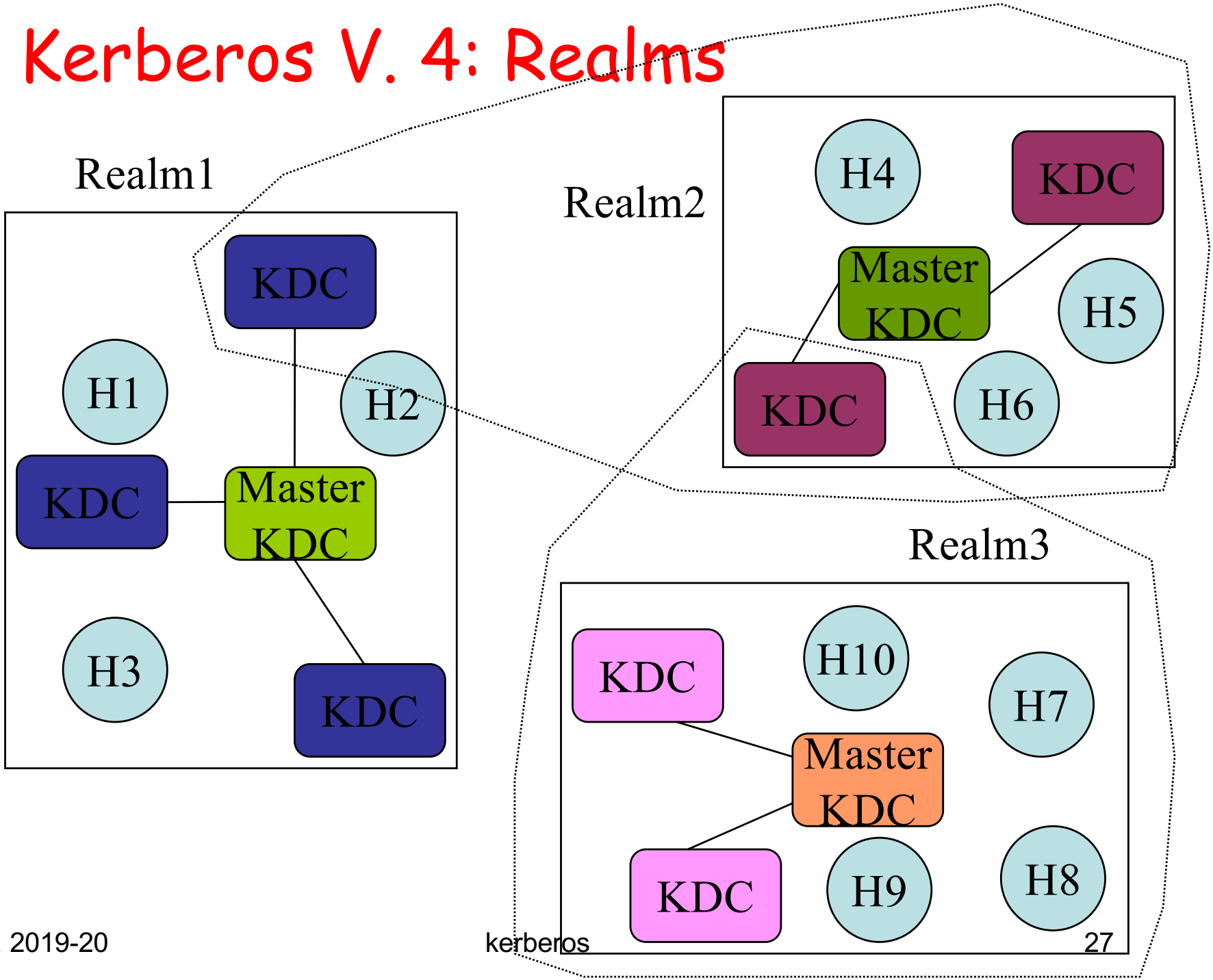
encrypted with session key

- Alice's name, instance and realm
- checksum
- 5-millisecond timestamp
- timestamp (time in seconds)
- pad of 0s to make authenticator multiple of eight octets

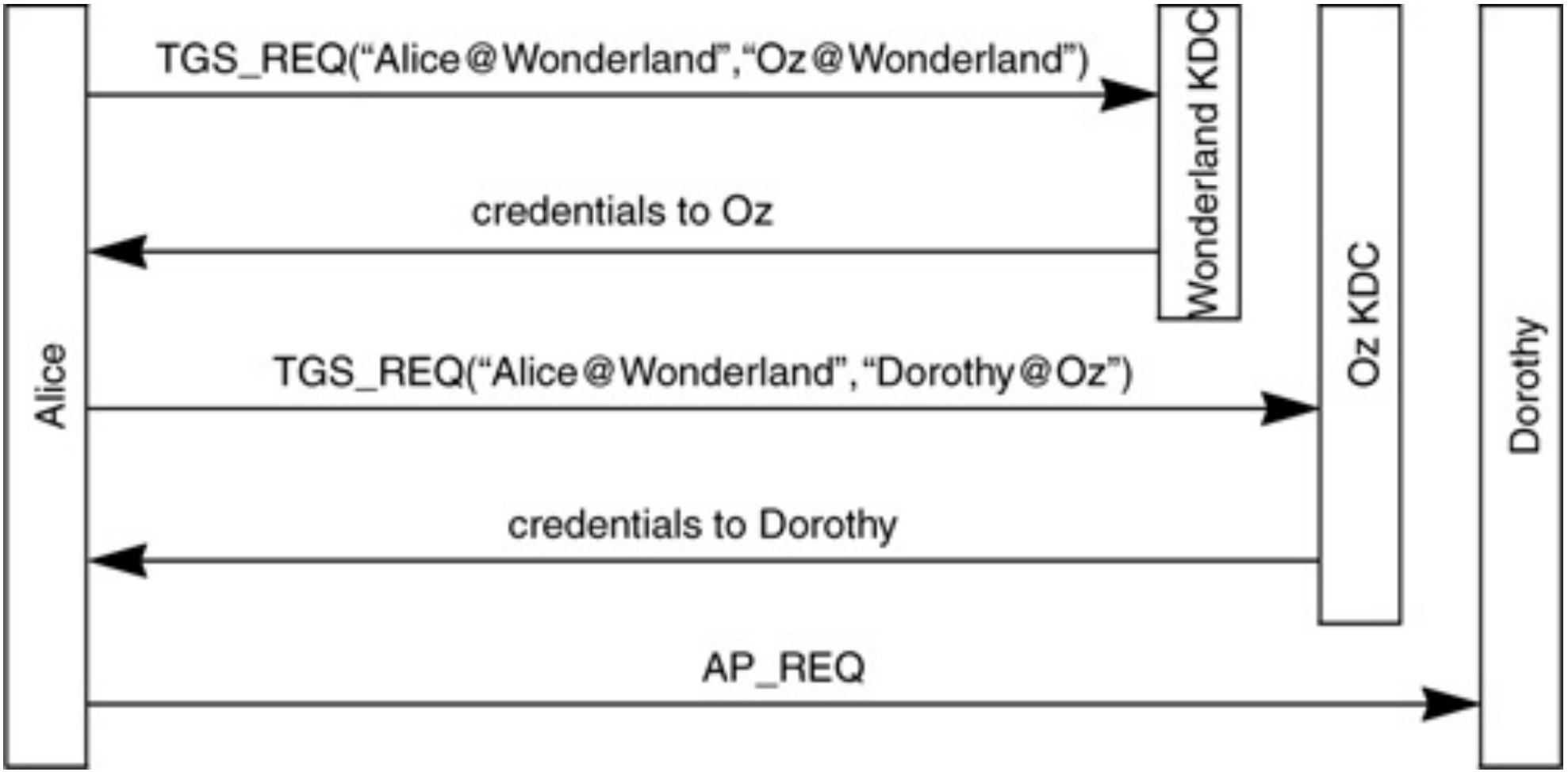
Kerberos: Realms

- 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

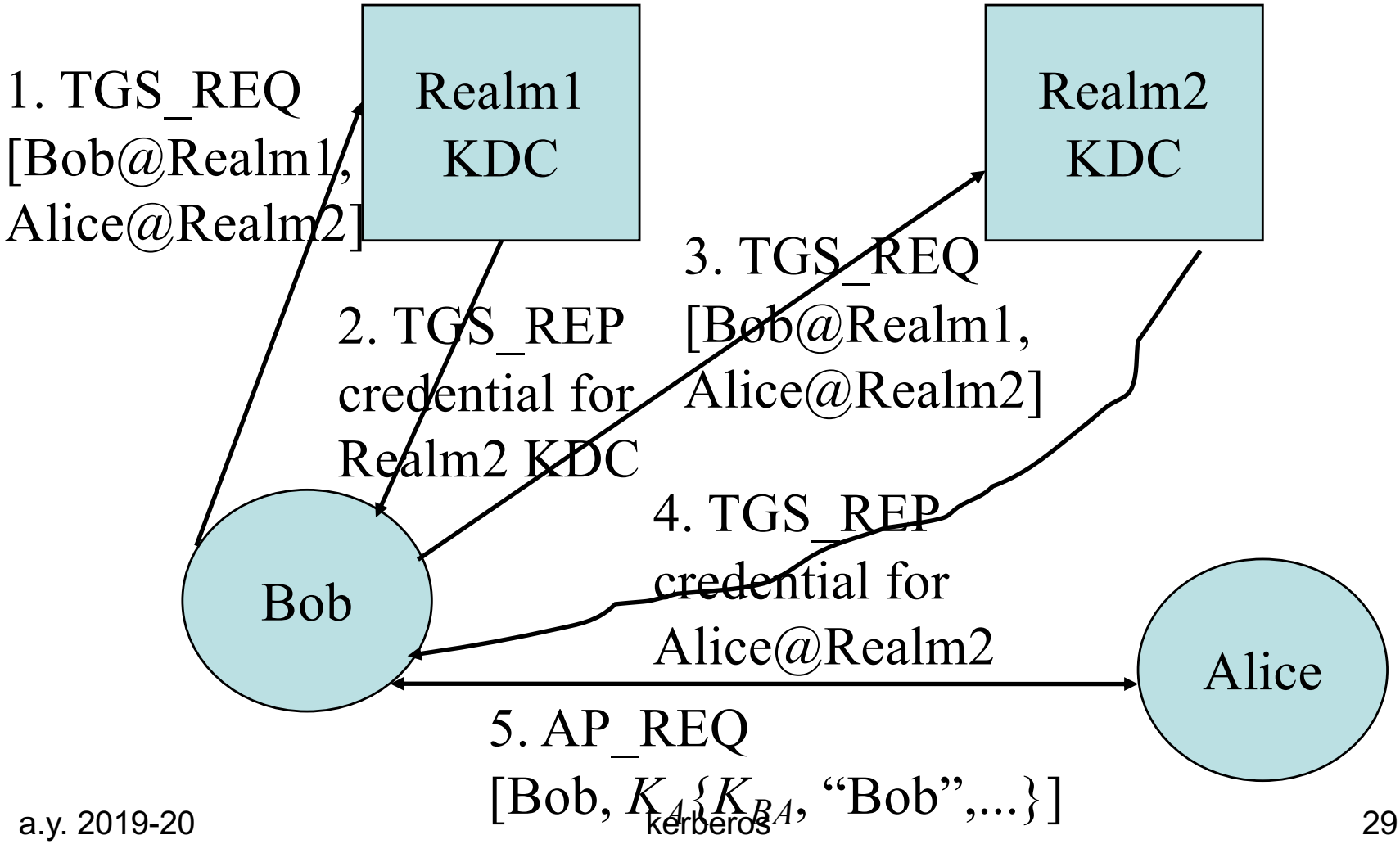
Kerberos V. 4: Realms



Authentication between realms



Authentication between realms



Other features

- key version numbers
  - for supporting changes of master keys
- encryption for privacy and integrity
  - DES + Plaintext Cipher Block Chaining
- 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