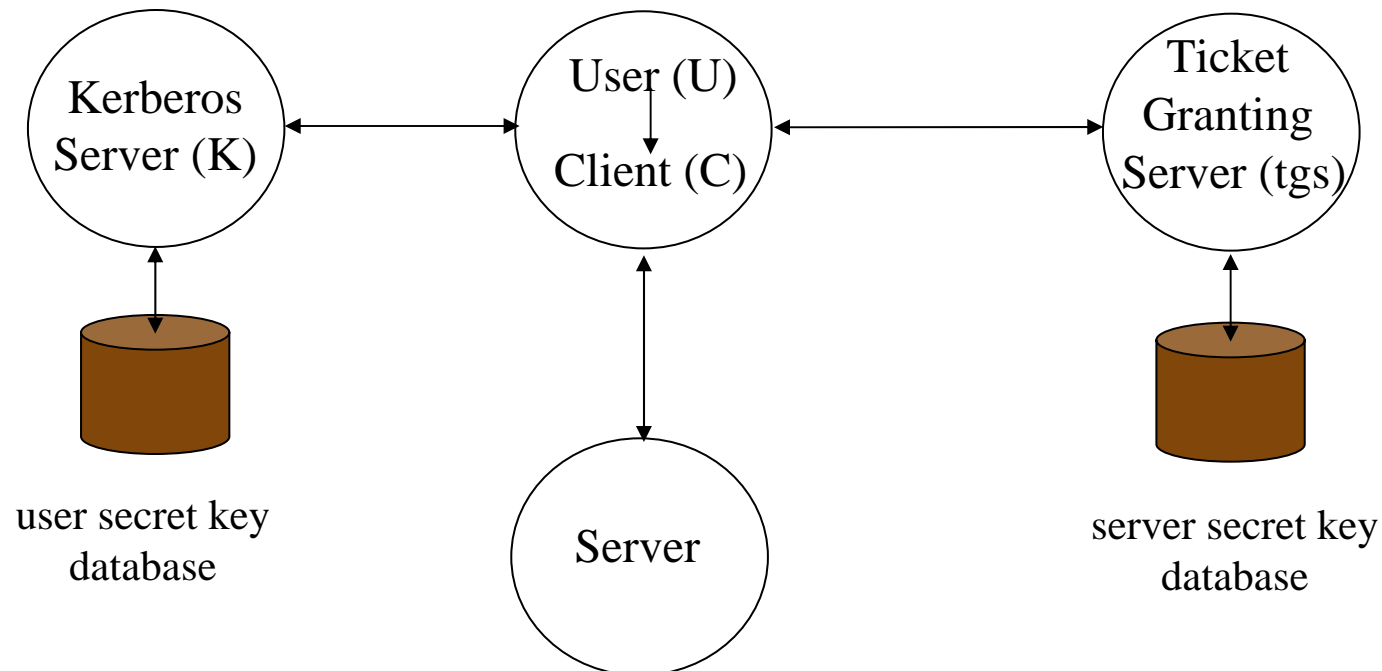


# A Private Key System

## KERBEROS

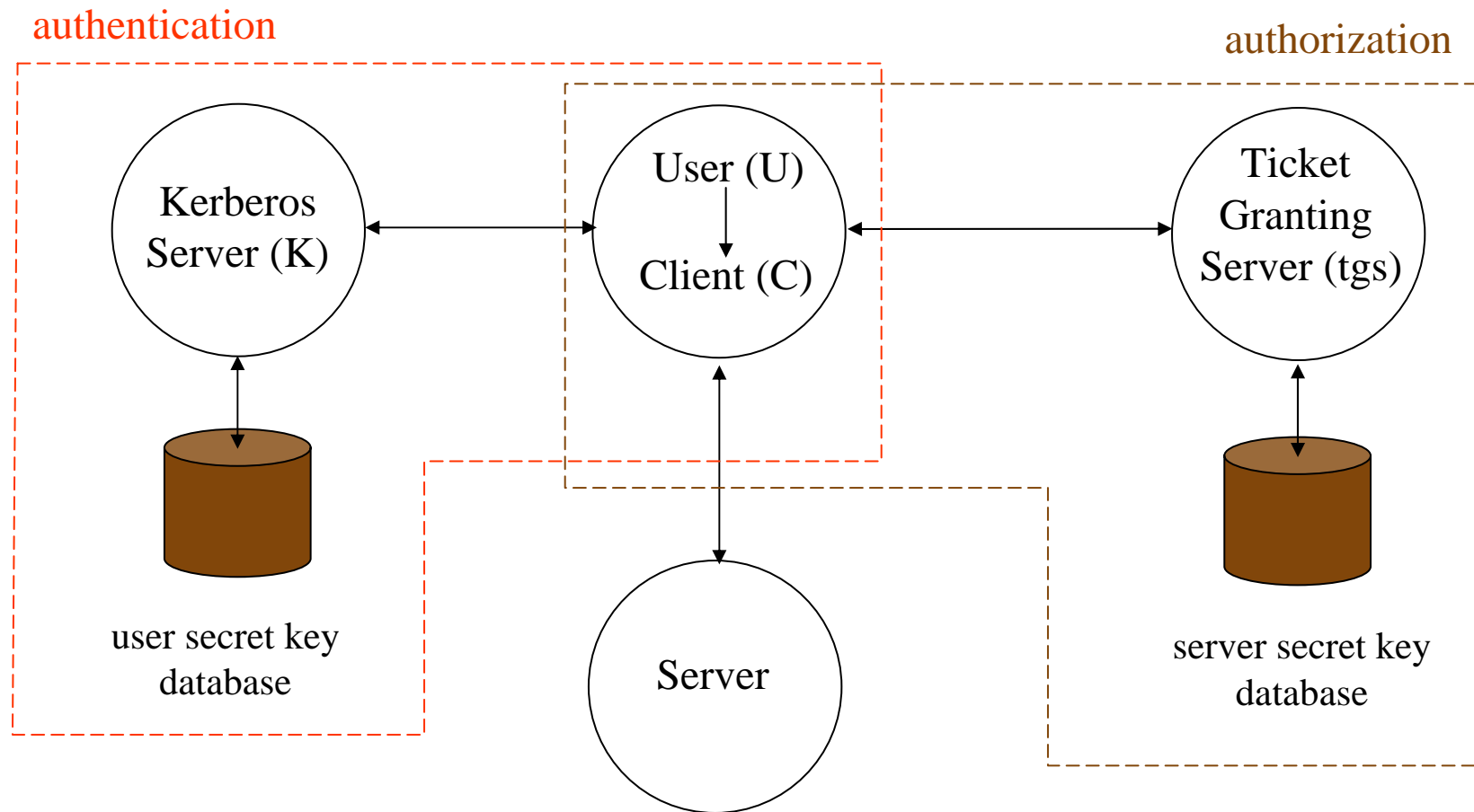
## Kerberos: Structure



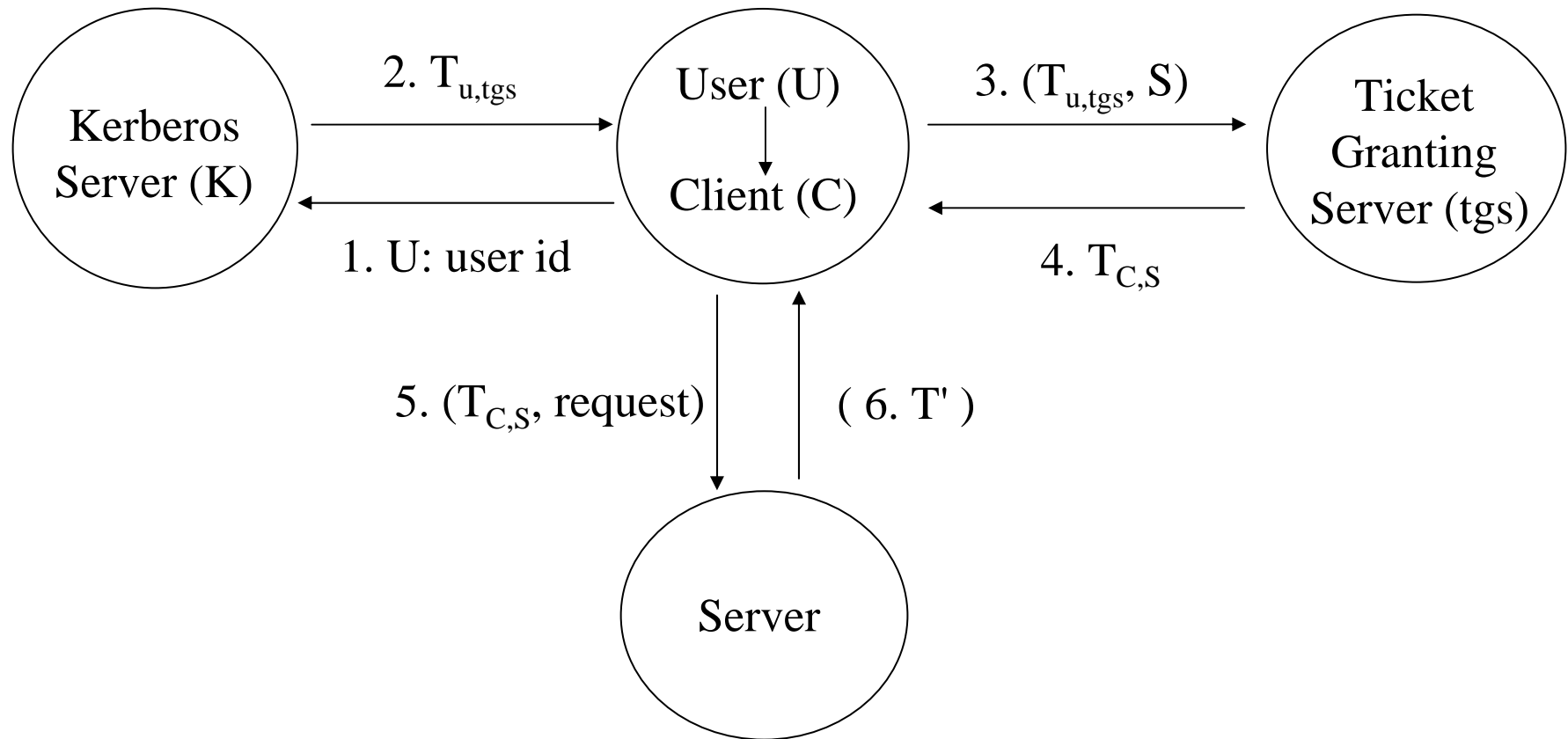
### Requirements:

- each user has a private password known only to the user
- a user's secret key can be computed by a one-way function from the user's password
- the Kerberos server knows the secret key of each user and the tgs
- each server has a secret key known by itself and tgs

# Kerberos: Steps



## Protocol Overview



### Ticket Structure:

$$E_{K(S)} \{ C, S, K_{C,S}, \text{timestamp}, \text{lifetime} \}$$

## Kerberos Phase 1

1. The user logs on to the client and the client asks for credentials for the user from Kerberos

$U \rightarrow C : U$  (user id)

$C \rightarrow K : (U, tgs)$

2. Kerberos constructs a ticket for  $U$  and  $tgs$  and a credential for the user and returns them to the client

$$T_{u,tgs} = E_{K(tgs)} \{ U, tgs, K_{U,tgs}, ts, lt \}$$

$K \rightarrow C : E_{K(U)} \{ T_{U,tgs}, K_{U,tgs}, ts, lt \}$

The client obtains the user's password,  $P$ , and computes:

$$K'(U) = f(P)$$

The user is authenticated to the client if and only if  $K'(U)$  decrypts the credential.

# Kerberos

## Phase 2

3. The client constructs an “authenticator” for user U and requests from TGS a ticket for server, S:

$$A_U = E_{K(U,tgs)} \{ C, ts \}$$

$$C \rightarrow TGS : (S, T_{U,tgs}, A_U)$$

4. The server authenticates the request as coming from C and constructs a ticket with which C may use S:

$$T_{C,S} = E_{K(S)} \{ C, S, K_{C,S}, ts, lt \}$$

$$TGS \rightarrow C : E_{K(U,tgs)} \{ T_{C,S}, K_{C,S}, ts, lt \}$$

# Kerberos

## Phase 3

5. The client builds an authenticator and send it together with the ticket for the server to S:

$$A_C = E_{K(C,S)} \{ C, ts \}$$
$$C \rightarrow S : (T_{C,S}, A_C)$$

6. The server (optionally) authenticates itself to the client by replying:

$$S \rightarrow C : E_{K(C,S)} \{ ts + 1 \}$$