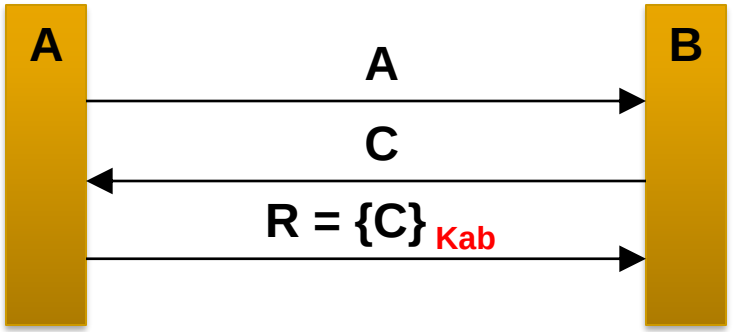
Authentication with Trusted Third Parties / KDCs

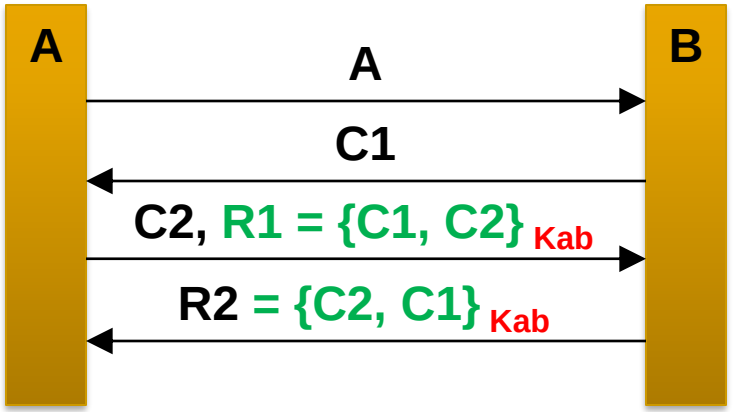
SAML Web Browser SSO Profile

Kerberos

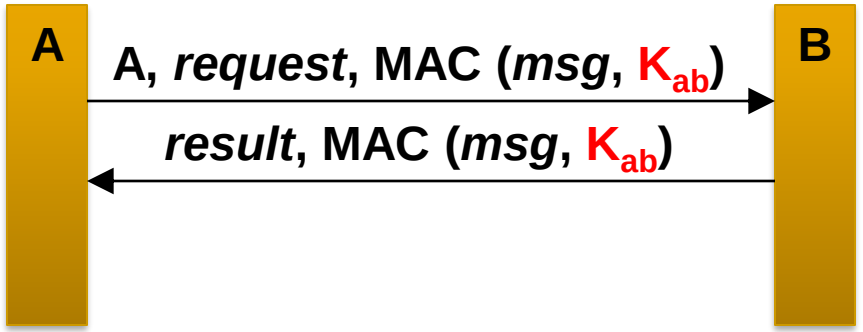
Shared-key authentication

- Connection-oriented





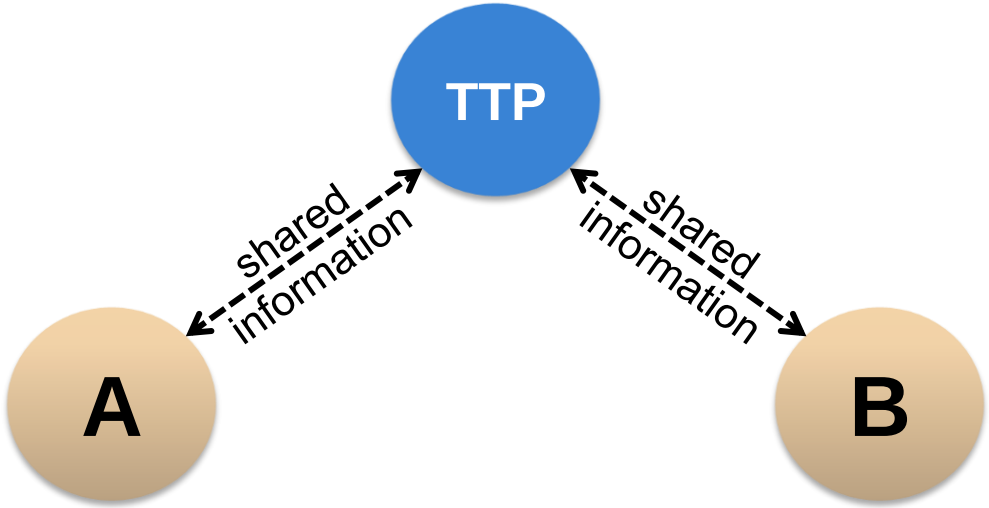
- Connection-less



- Issue
  - How to distribute K_{ab} to all possible A-B pairs?

Authentication with Trusted Third Party: Key Distribution Center (KDC) concept

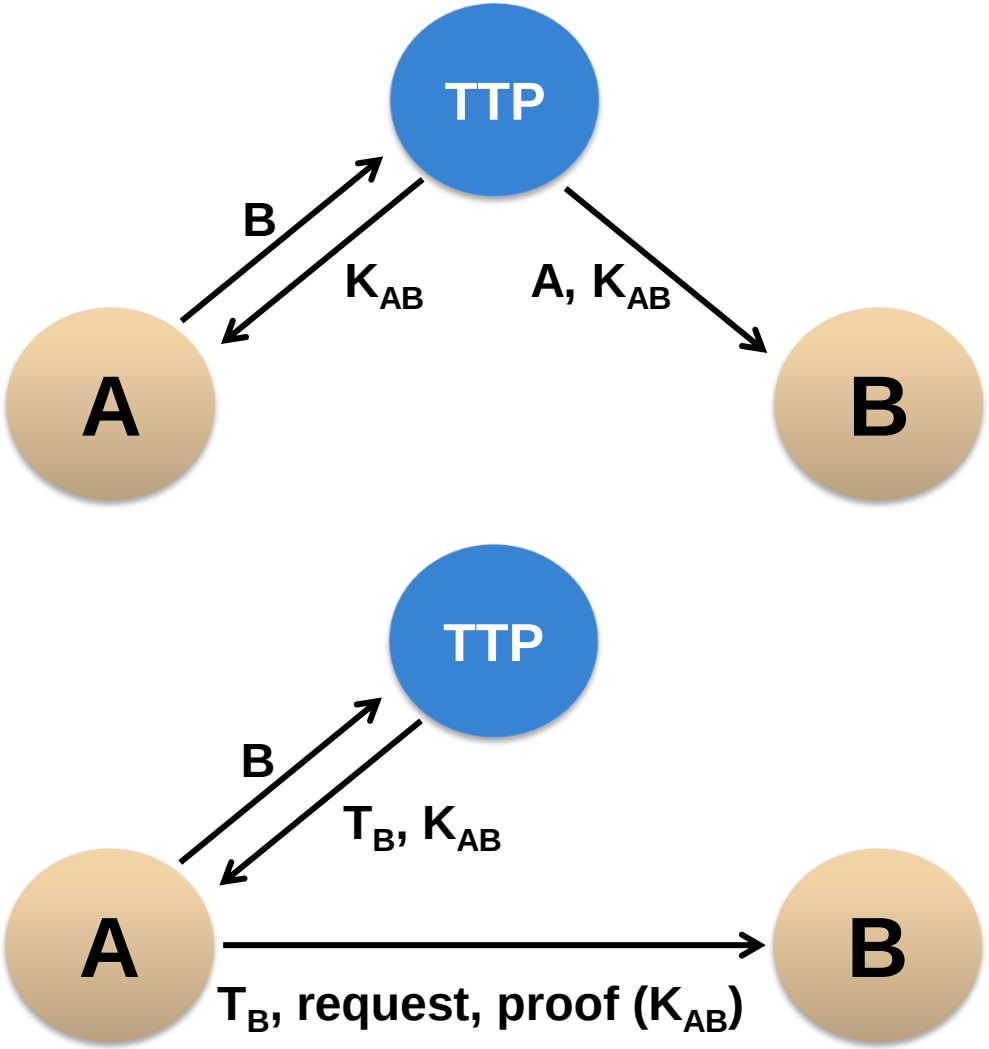
- TTP is responsible for bridging the gap between peers
  - A and B don't have any shared information
  - A and B have shared information with TTP



Why KDC?

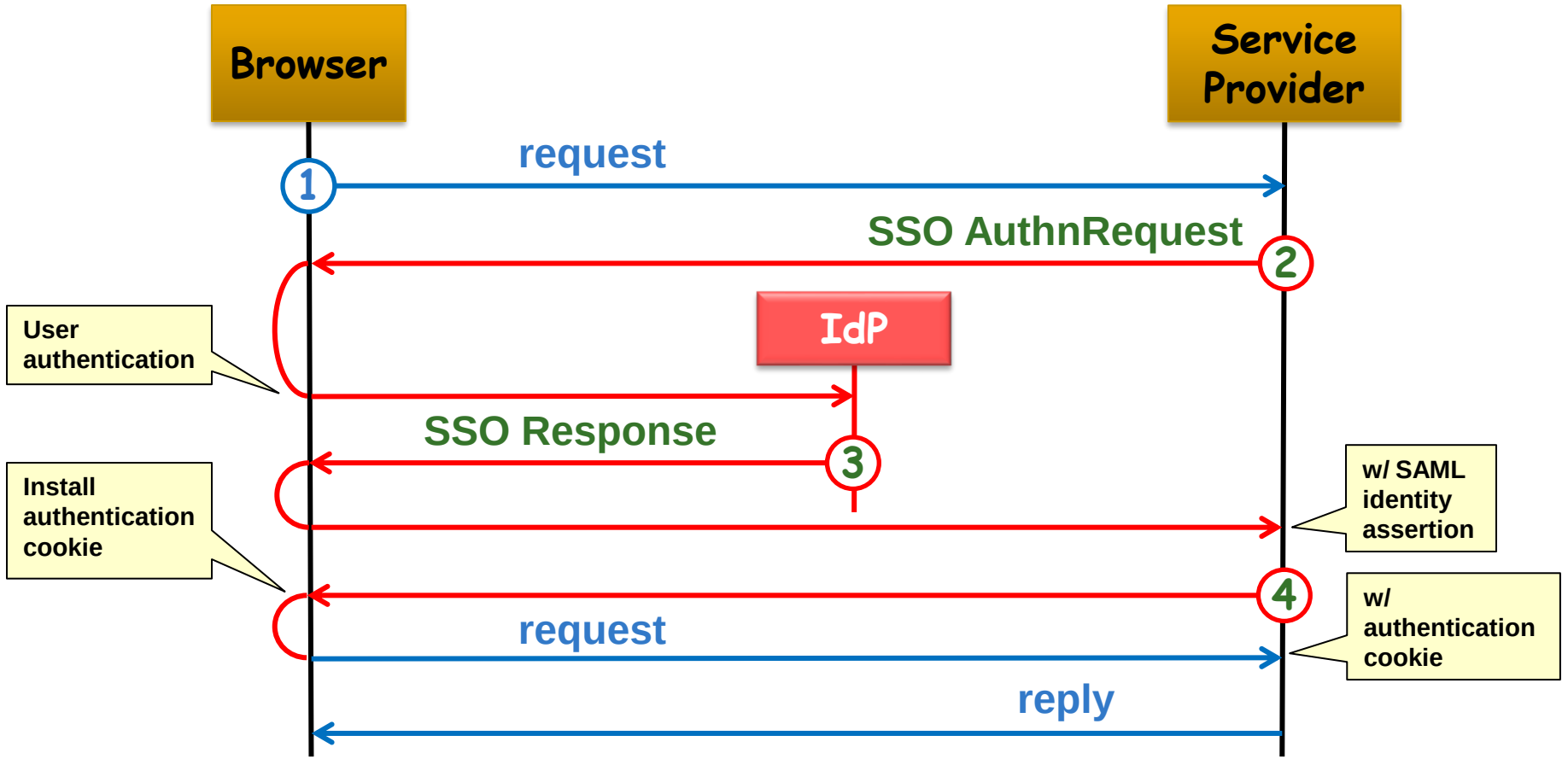
- Because a TTP can distribute a session key to A and B for proving each other their identity
  - Session key K_{AB}
    - It is temporary (only for one session)
  - A uses K_{AB} to prove its identity is B
  - B uses K_{AB} to prove its identity is A
- The proofs by A and B can be made in different ways
  - Only in the beginning of a session
  - On each interaction along a session

Session key distribution



Example:

SAML Web Browser SSO Profile

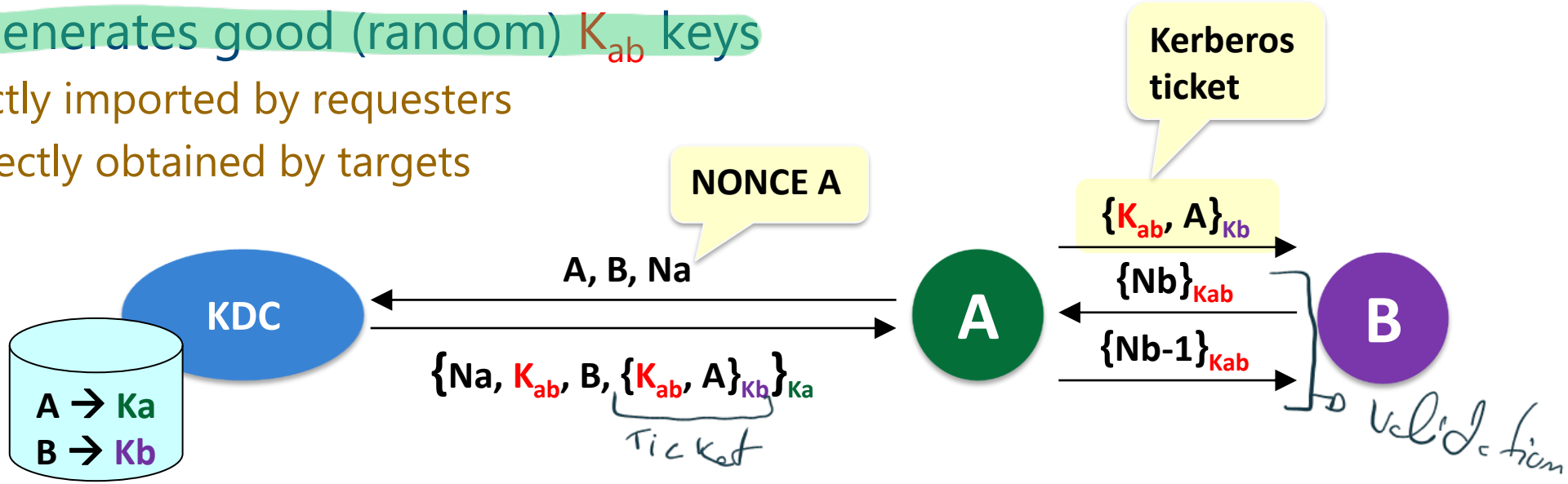


Kerberos: Goals

- Authenticate peers in a distributed environment
  - Targeted for Athena (at MIT)
- Distribute session keys for adding security to sessions between peers
  - Authentication (the initial goal)
  - Confidentiality (optional)
- Single Sign-On
  - Only one password to remember
  - Daily use (typically)

Kerberos background: Needham-Schroeder (1978)

- A and B trust on a common KDC
  - Key Distribution Center
- KDC shares a key with every A and B
  - Central authentication authority
- KDC generates good (random) K_{ab} keys
  - Directly imported by requesters
  - Indirectly obtained by targets



Kerberos:
Architecture and base concepts

- Architecture
  - Two Kerberos KDC services
    - Authentication Service (AS)
    - Ticket Granting Server (TGS)
  - Entities (principals)
    - All have a secret shared with Kerberos (AS or TGS)
    - People: a key derived from a password:
      K_U = hash(password)
    - Services/servers: key stored in some repository
  - Requisites
    - Clocks (very well) synchronized
- Authentication elements
  - Ticket: required to make a request of a service
  - Authenticator: proof of the identity of a requester

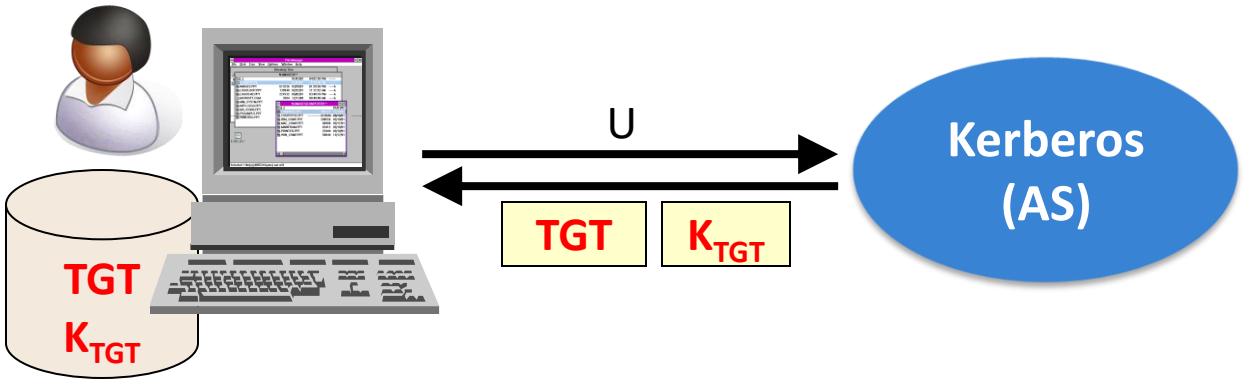
Kerberos:
Tickets and authenticators

- Ticket
  - Unforgeable piece of data
  - Can only be interpreted by the target service
  - Carries the identities of the client that can use it
  - Carries a session key
  - Carries a validity timestamp

- Authenticator
  - Carries a timestamp of the request
  - Carries the identity of the client
  - Proves that the client knows the session key

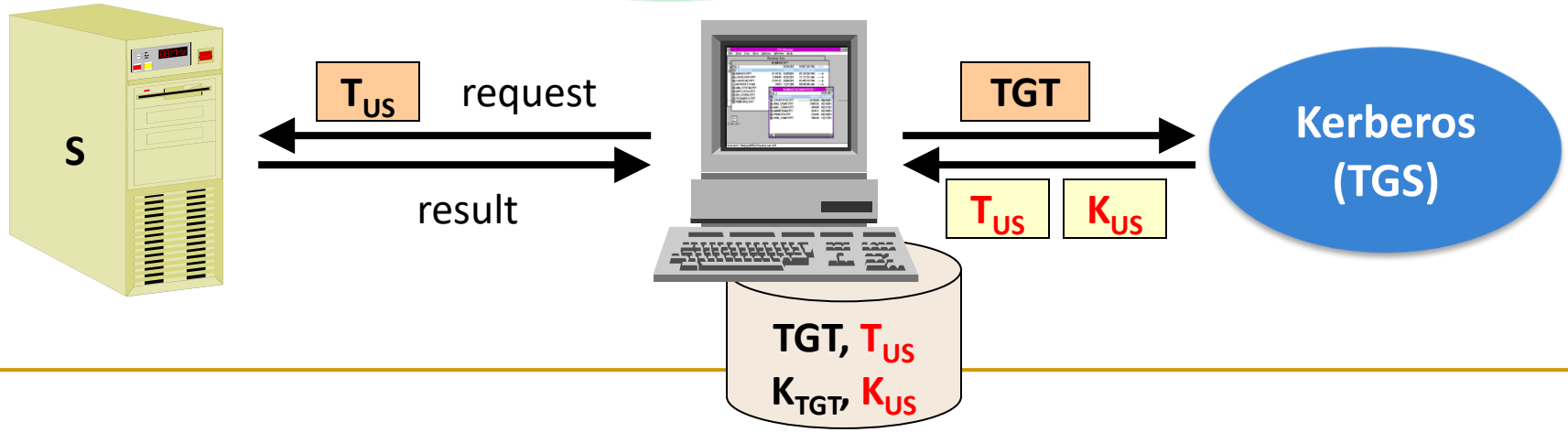
Overview of Kerberos SSO: 1st step: Login

- Location of the Kerberos servers of the realm
- Authentication of user U by Kerberos (AS)
  - User gets a Ticket Granting Ticket (TGT) and a session key (K_{TGT}) for interacting with another Kerberos service (TGS)
  - The TGT can be used to request other tickets needed by the user U to access each and every service S

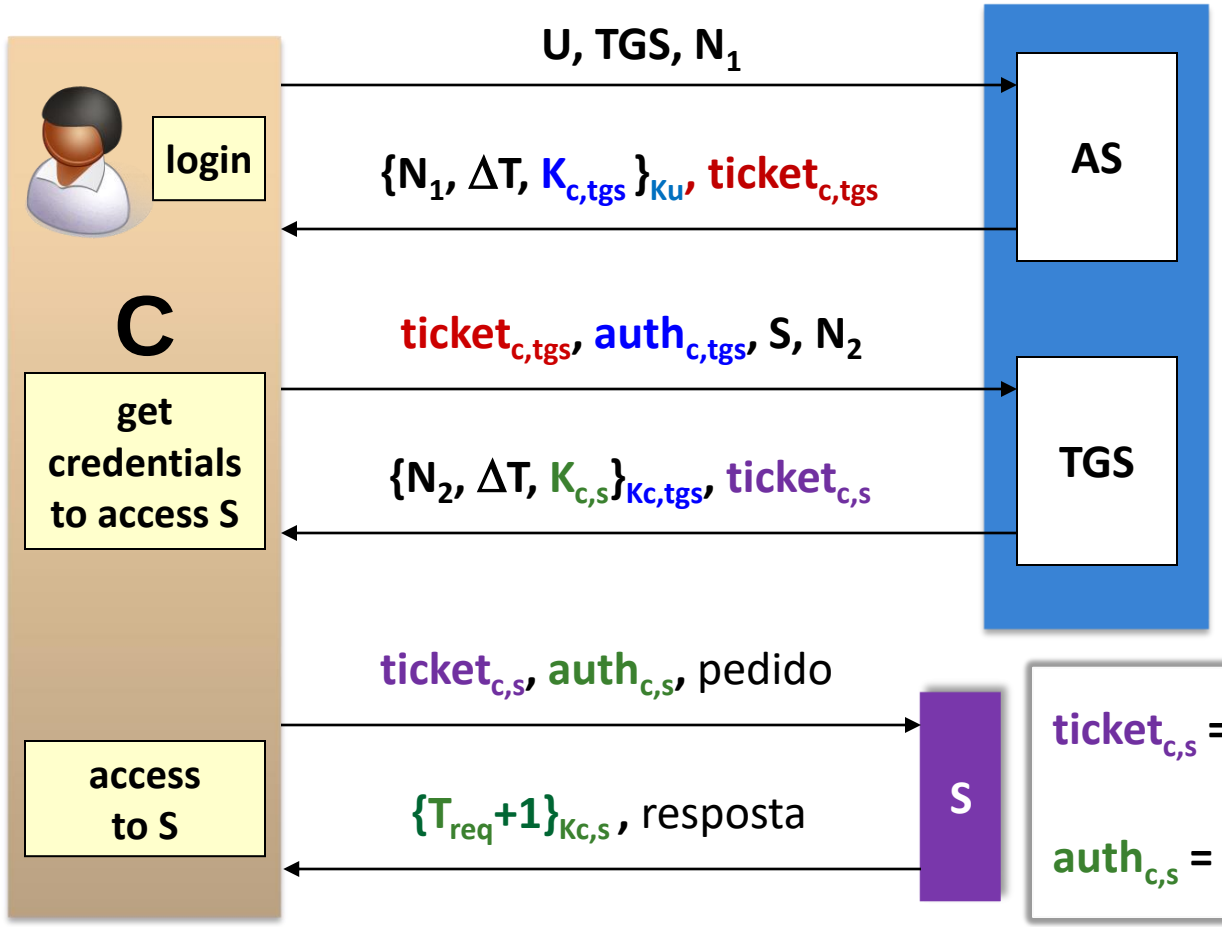


Overview of Kerberos SSO: 2nd step: Authenticated access to servers

- U requests Kerberos (TGS) a ticket for accessing S
  - U uses TGT in the request
  - U must prove that he is the owner of TGT
  - U gets a session key (K_{US}) and a ticket to S (T_{US})
- U uses T_{US} to make authenticated requests to S
  - Server S uses T_{US} to check the identity of U
  - U must prove that he is the owner of T_{US}



Kerberos:
Protocol (of version V5)

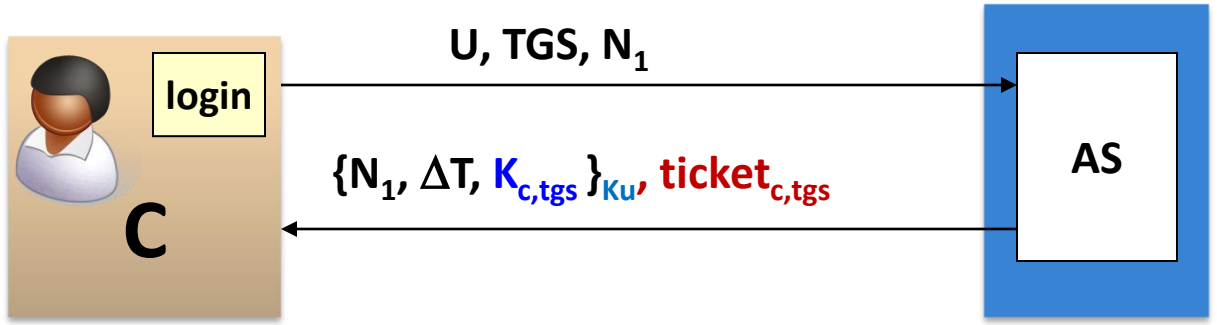


	C (U)	TGS	S
AS	K_U	K_{tgs}	
TGS	$K_{c,tgs}$		K_s
S	$K_{c,s}$		

$ticket_{c,s} = S, \{U, addrs_c, \Delta T, K_{c,s}\}_{Ks}$

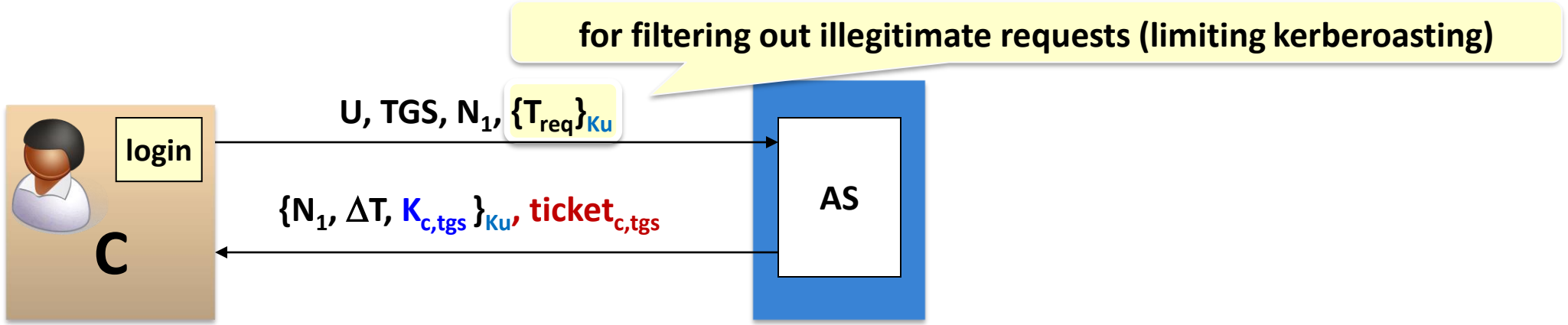
$auth_{c,s} = \{U, T_{req}, K', seq\}_{Kc,s}$

Kerberos:
Pre-authentication alternative



	C (U)	TGS	S
AS	K_U	K_{tgs}	
TGS	$K_{c,tgs}$		K_s
S	$K_{c,s}$		

- Vulnerable to proactive dictionary attacks! (Kerberoasting)



Kerberos: Scalability

- Authentication scope
  - Realms
  - A kerberos server per realm
- Inter-realm cooperation
  - Fundamental to allow a client from a realm to access a server on another realm
  - Realms need to trust on authentication performed by other realms
- Protocol
  - Secret keys shared between TGS servers of different realms
    - Inter-realm key
    - Each inter-realm key is associated to a trust path
  - A client (user) needs to jump from TGS to TGS for getting a ticket
    - Not particularly user-friendly

Kerberos V5:
Security politics and mechanisms

- Entity authentication
  - Secret keys, names, networks addresses
  - name/instance@realm (user@ua.pt, ftp/ftp.ua.pt@ua.pt)
- Validity periods
  - Timestamps in tickets (hours)
  - Timestamps in authenticators (seconds, minutes)
- Replay protections
  - Nonces (in ticket distributions)
  - Timestamps / sequence numbers (in authenticators)
- Protection against an excessive use of session keys
  - Key distribution in authenticators
- Delegation (proxying)
  - Options and authorizations in tickets
- Inter-real authentication
  - Secret keys shared among TGS services, trust paths
  - Ticket issuing from a TGS to another TGS

Kerberos: Security issues

- Kerberos KDC can impersonate anyone
  - Needs maximum security in its administration
- Kerberos KDC may be a single point of failure
  - Replication is an option, since stored keys are seldom updated
- A stolen user password allows others to impersonate the victim in every service of the realm
  - Stolen TGS credentials are less risky, as their validity is shortly limited (≈ one day, usually)

Kerberos V5: Actual availability

- MIT releases
  - http://web.mit.edu/kerberos
  - Sources and binaries
- Windows versions
  - Windows 2000 adopted Kerberos for inter-domain authentication
  - Kerberos was modified to accommodate Windows credentials
- Components
  - Kerberos servers/daemons
  - Libraries for “kerberizing” applications
  - Support applications
    - klogin, kpasswd, kadmin
  - Kerberized applications (clients and servers)