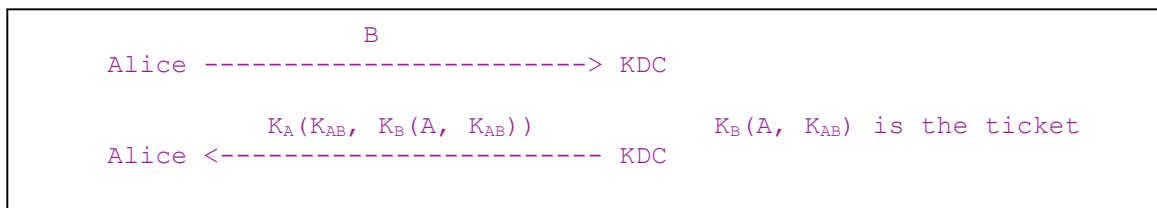


Kerberos

- ❖ Key Distribution
 - Public Key (using certificate)
 - Symmetric key: How to distribute symmetric keys between two communicating parties?
- ❖ Symmetric Key Distribution
 - Using Diffie-Hellman
 - Using Public-Key Cryptography (SSL)
 - Using Trusted 3rd Party
- ❖ Kerberos
 - Originally designed at MIT
 - KDC: Key Distribution Center. It is also called Authentication Server (AS). KDC runs on a physically secure node
 - AS shares a secret key (master key) with each principal
- ❖ Alice wants to talk to Bob



- ❖ How to derive Alice's master key?
 - login: From Alice's password
 - Q: If Alice later wants to talk to C, D, and E, each time, either Alice has to retype the password, or the computer has to store Alice's master key in memory. In the latter case, it is not secure (some other programs can read the memory and get the key). **How to solve this problem?**
- ❖ Ticket-granting ticket (TGT)
 - Before accessing any regular service, the user first gets a ticket-granting ticket that allows her to talk to TGS (Ticket Granting Server).
 - After receiving the TGT, any time that the user wishes to contact a service, she requests a ticket not from the AS, but from the TGS.
 - Furthermore, the reply from the TGS is encrypted not with the user's secret key, but with the *session key* (S_A) that the AS provided for use with the TGS. Inside that reply is the new session key for use with the regular service.
 - *The AS and the TGS are logically distinct but are usually physically identical (same process).*

```

      KA(SA, KTGS(A, SA, expiration time))
Alice <----- KDC

      B, KTGS(A, SA, expiration time), authenticator = SA{timestamp}
Alice -----> TGS

      SA(KAB, KB(A, KAB))
Alice <----- TGS

```

- Bob authenticates Alice.

```

      KB(A, KAB), authenticator = KAB(timestamp)
Alice -----> Bob

```

- Alice authenticates Bob.

```

      KAB(timestamp + 1)
Alice <----- Bob

```

❖ Realm

- Principals in the network are divided into *realms*
- These divisions are often made on organizational boundaries, although they need not be. Each realm has its own AS and its own TGS.
- How to access a principal in another realm? i.e., how to allow users in one realm to access services in another?

❖ Cross Realm Authentication

- The user's realm needs to register a *remote* TGS (RTGS) in the service's realm.
- An additional exchange is added to the protocol: First, the user contacts the AS to access the TGS. Then the user contacts the TGS to access the RTGS. Finally, the user contacts the RTGS to access the actual service.
- For example, Alice in realm Wonderland wishes to talk to Dorothy in realm Oz.
 - Oz's KDC must register itself as a principal of Wonderland's KDC.
- When there are many realms, it is inefficient to register each realm in every other realm. How to solve this scalability issue?
 - There is a hierarchy of realms, so that in order to contact a service in another realm, it may be necessary to contact the RTGS in one or more intermediate realms. The names of each of these realms is recorded in the ticket.
 - This feature is new to Kerberos in version 5. In version 4, only peer-to-peer cross-realm authentication was permitted. In other words, in an environment with 100 realms, complete authentication coverage required the registration of $100 \cdot 99 = 9900$ remote TGS.

