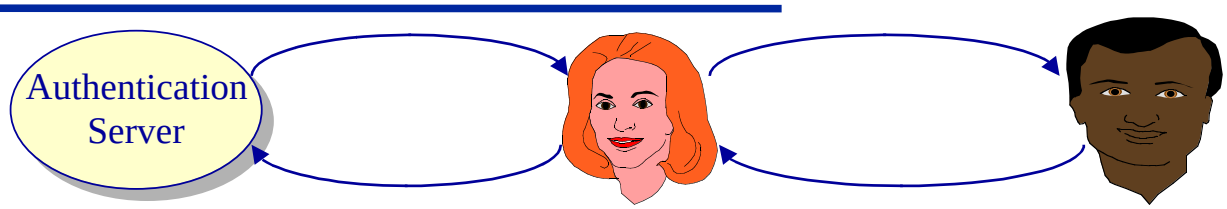
COMP 243
Distributed Systems

The Kerberos Authentication System

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
October 27, 1999

http://www.cs.unc.edu/~jeffay/courses/comp243f99

The Kerberos Authentication System
Overview

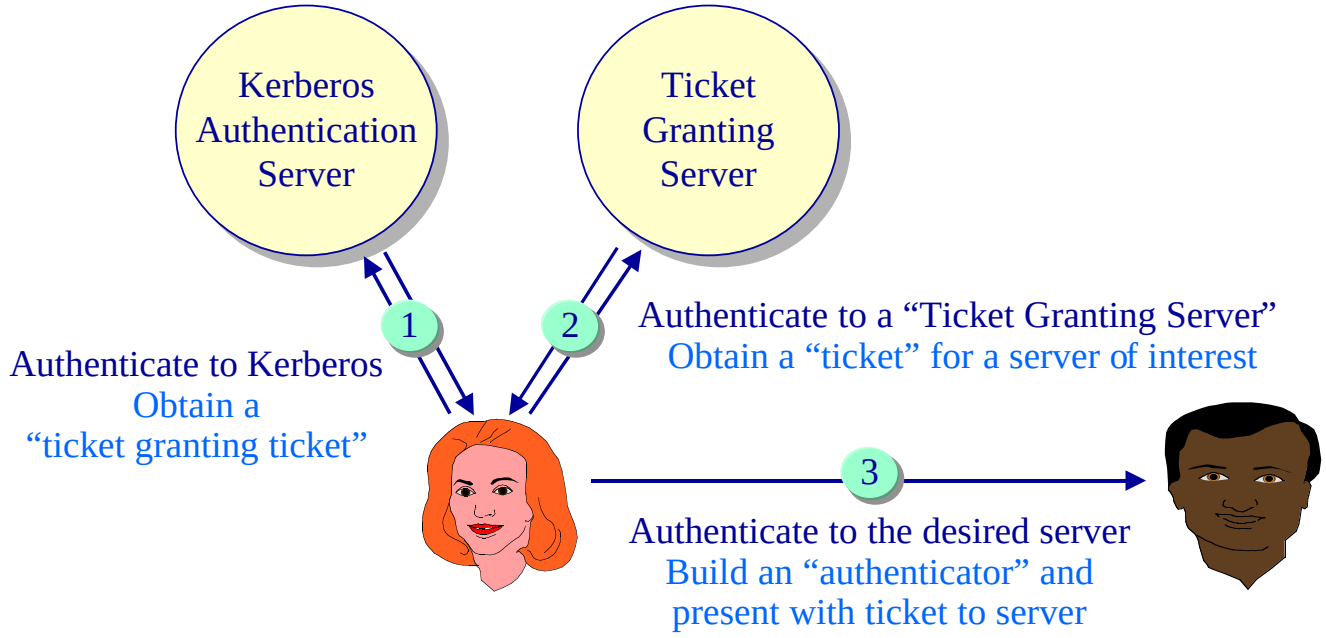


- An implementation of the Needham & Schroeder protocol
  - » Encryption is based on DES
  - » Timestamps added to defend against replay attacks
- Used in OSF Distributed Computing Environment (DCE) and separately in AFS
- Basic services:
  - » Authentication at time of connection establishment
  - » “Safe (authenticated) messages”
  - » “Private (encrypted) messages”

Authentication in Kerberos

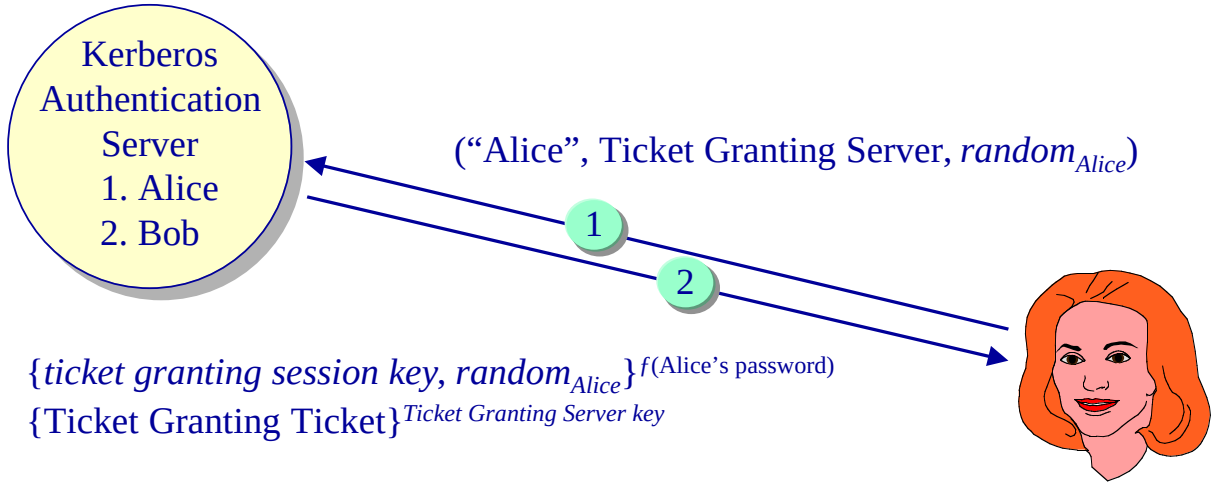
A three step process

- User authentication and client/server session key generation functions separated



Authentication in Kerberos

Step 1 — Authenticating to Kerberos

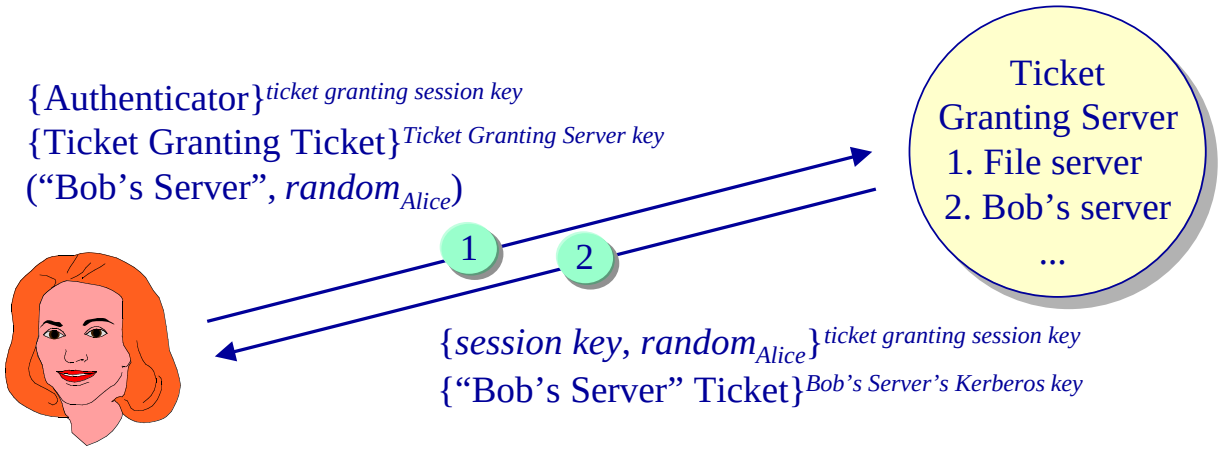


Alice's Ticket Granting Ticket:

1. "Alice"
2. "Ticket Granting Service"
3. *ticket timestamp*
4. *expiration date*
5. *Alice's IP address*
6. *ticket granting session key*

Authentication in Kerberos

Step 2 — Authenticating to the Ticket Granting Server



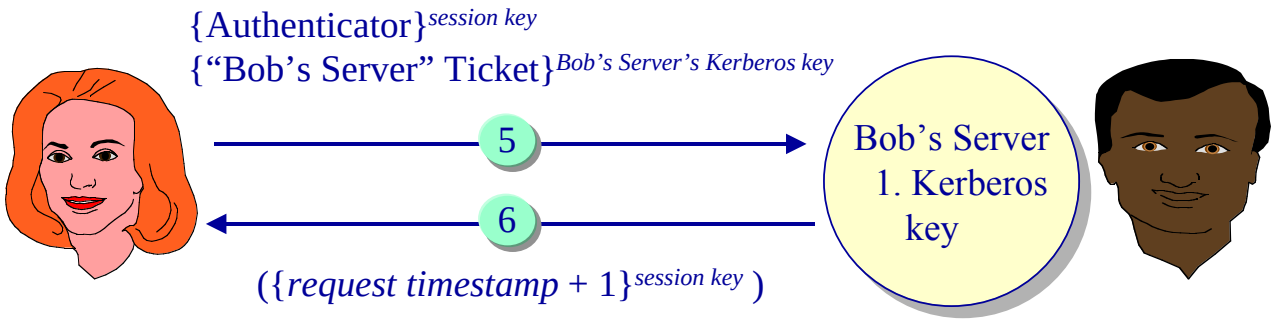
Alice's Ticket Granting Ticket:
1. "Alice"
2. "Ticket Granting Service"
3. *ticket timestamp*
4. *expiration date*
5. Alice's IP address
6. *ticket granting session key*

Alice's Authenticator:
1. "Alice"
2. *request timestamp*
3. Alice's IP address

Alice's "Bob's Server" Ticket:
1. "Alice"
2. "Bob's Server"
3. *ticket timestamp*
4. *expiration date*
5. Alice's IP address
6. *session key*

Authentication in Kerberos

Step 3 — Authenticating to Bob's server

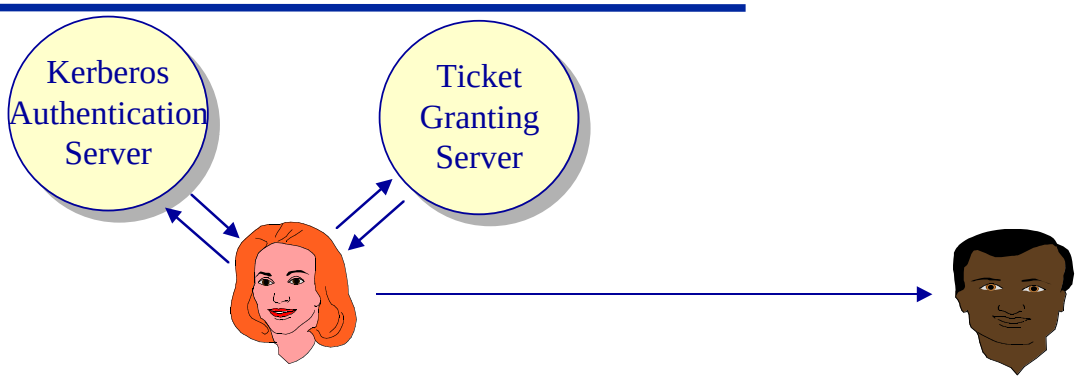


Alice's "Bob's Server" Ticket:
1. "Alice"
2. "Bob's Server"
3. *ticket timestamp*
4. *expiration date*
5. Alice's IP address
6. *session key*

Alice's One-Time Authenticator:
1. "Alice"
2. *request timestamp*
3. Alice's IP address

Authentication in Kerberos

Kerberos as a distributed service

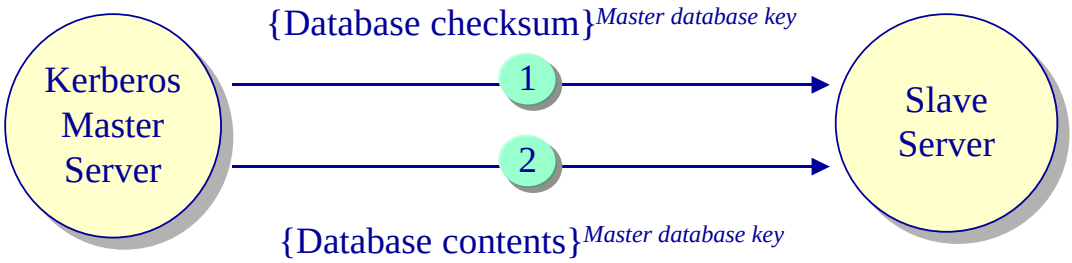


- Scalablity
  - » Hierarchical authentication as in the DNS
- Fault tolerance
  - » Replicated key database
- Transparency
  - » Existing programs easily adapted to use Kerberos

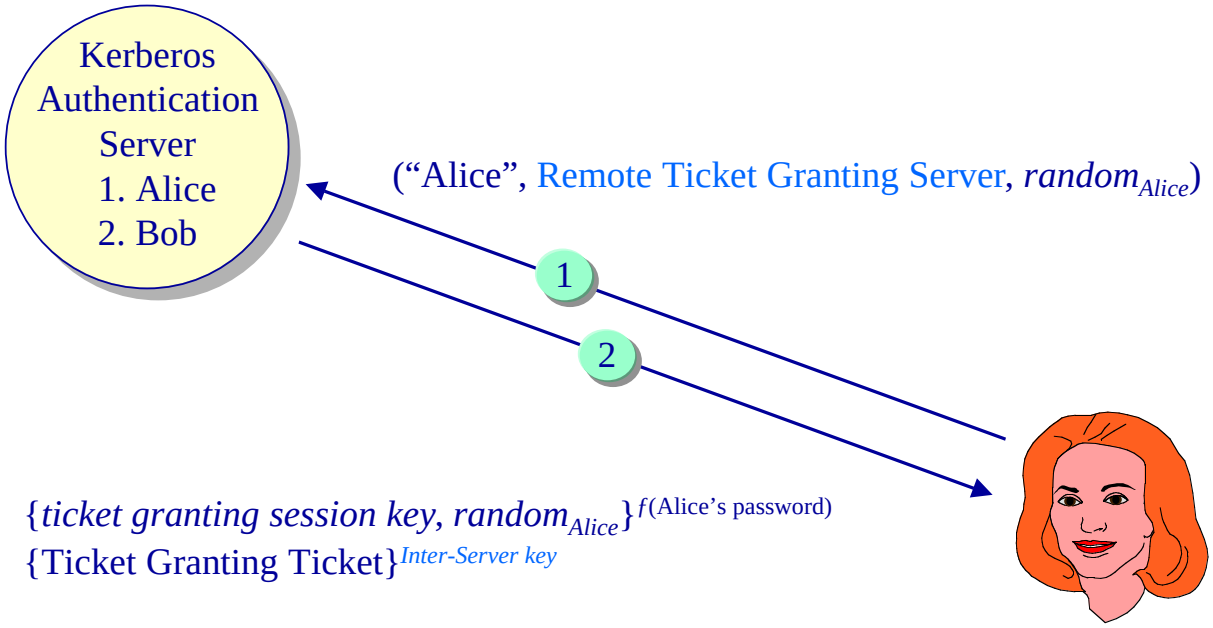
Kerberos as a Distributed System

Scalability

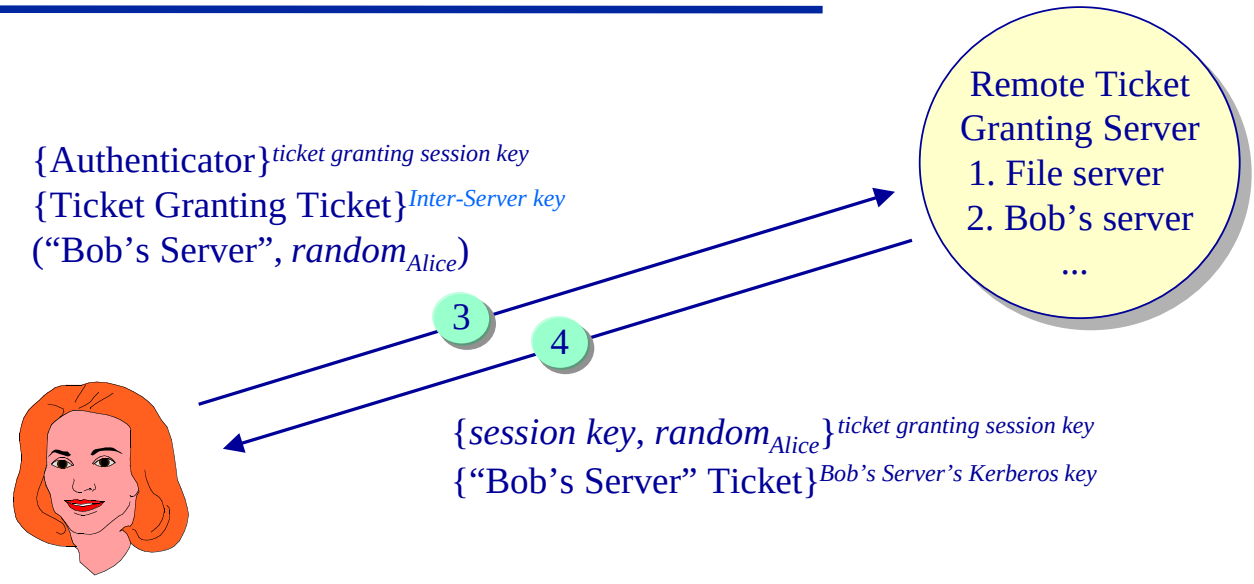
- Replicate Kerberos servers
  - » 1 master & *n* read-only, slave databases
- Replicas updated every hour
  - » Entire contents of database sent to every slave



Scalability
Inter-domain authentication



Scalability
Inter-domain authentication

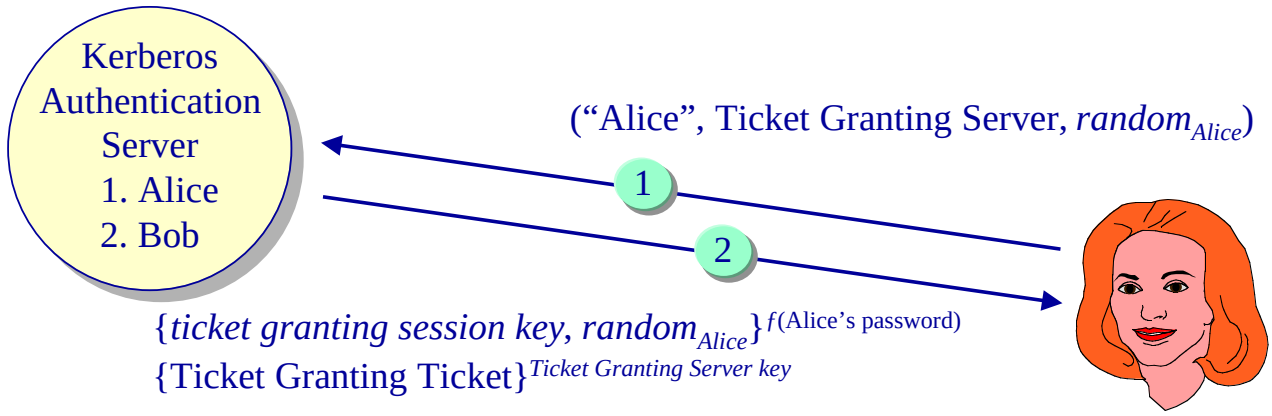


Or for transitive authentication:

3*a* {Authenticator}$^{ticket\ granting\ session\ key}$
{*path*, Ticket Granting Ticket}$^{Inter\text{-}Server\ key}$
(Intermediate server, $random_{Alice}$)

Kerberos as a Distributed System

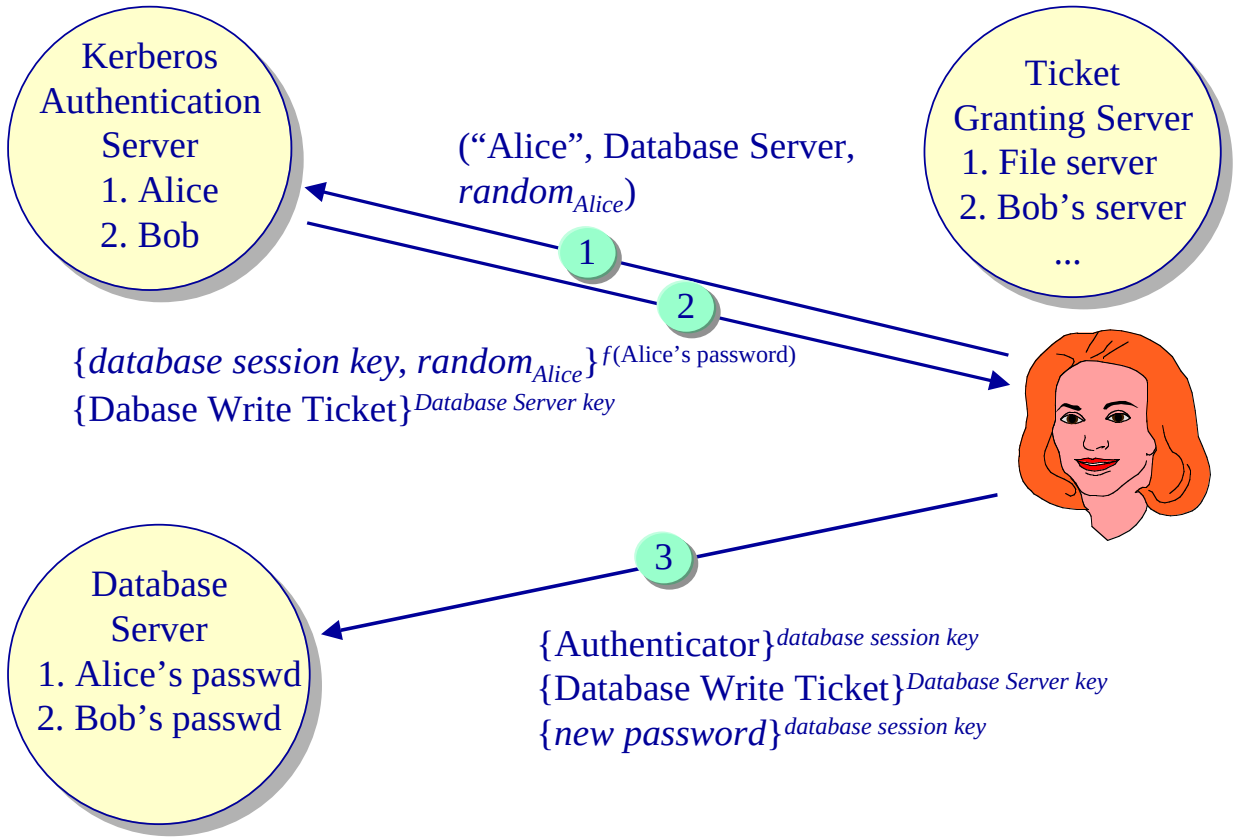
Transparency



- Authentication occurs when users log-in
  - » Users enters name
  - » Message exchanges 1 & 2 occur
  - » If user is known, she is prompted for her password
  - » Password converted to a DES key & used to decrypt message #2
  - » If successful, ticket is saved, password & DES key are erased

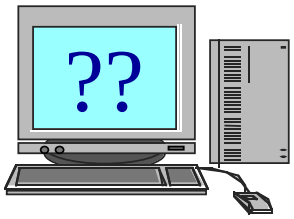
Transparency

Updating the Kerberos database

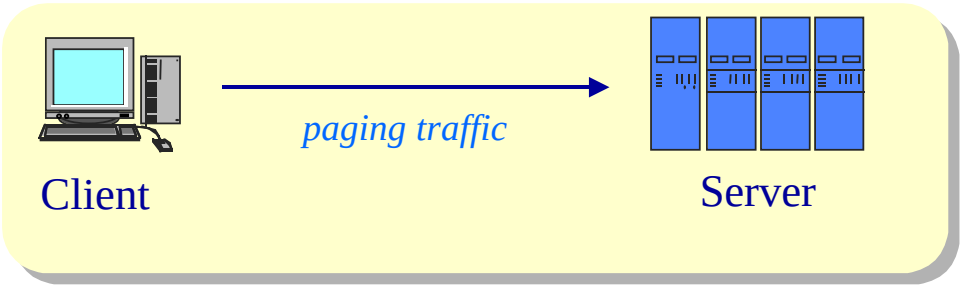


Weaknesses of Kerberos

Security of encryption keys

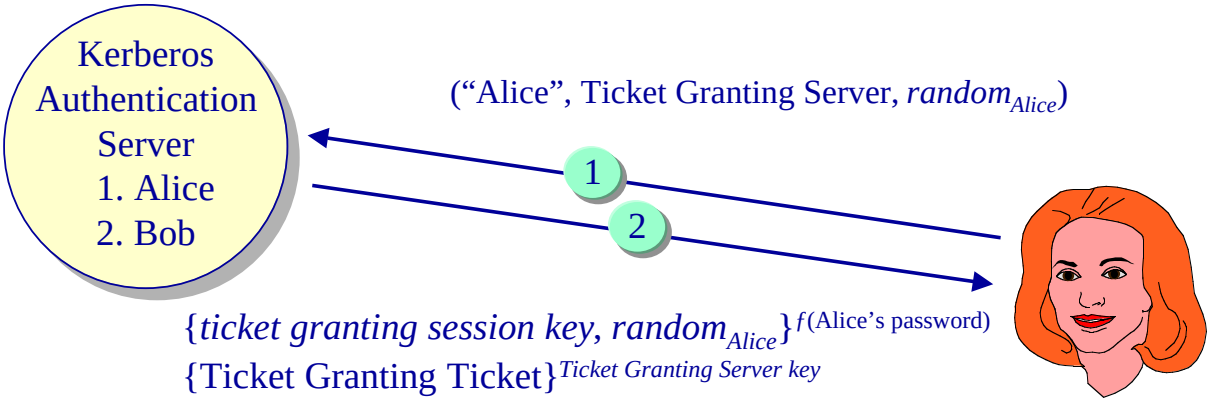


- Can encryption keys be securely stored on a workstation?
  - » How secure is `/dev/kmem`?
  - » What if a machine has no local `/tmp` or swap space?



Weaknesses of Kerberos

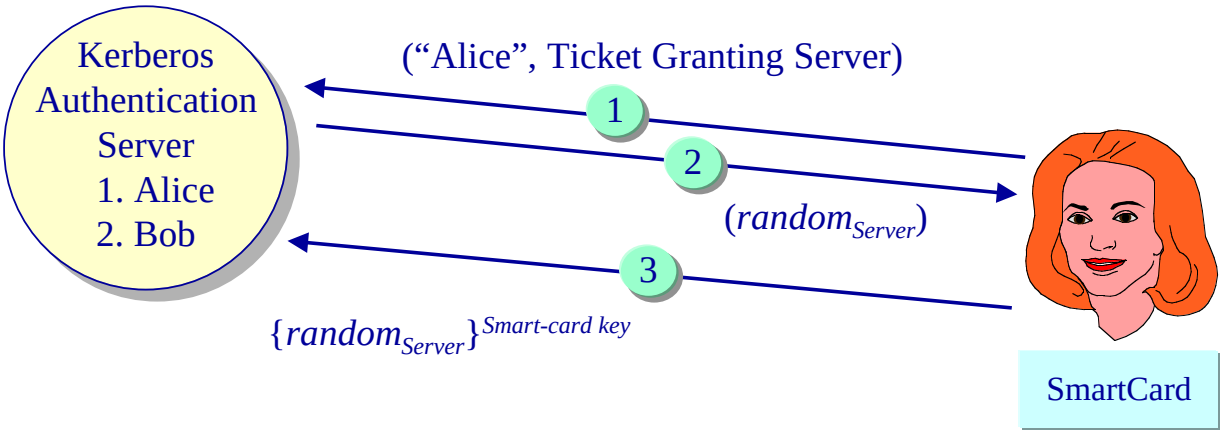
Security of user passwords



- Is it really easy to guess a user's password?
- `login` spoofing attacks
  - » Physical security of the end-system cannot be guaranteed

Avoiding Password Guessing/Spoofing Attacks

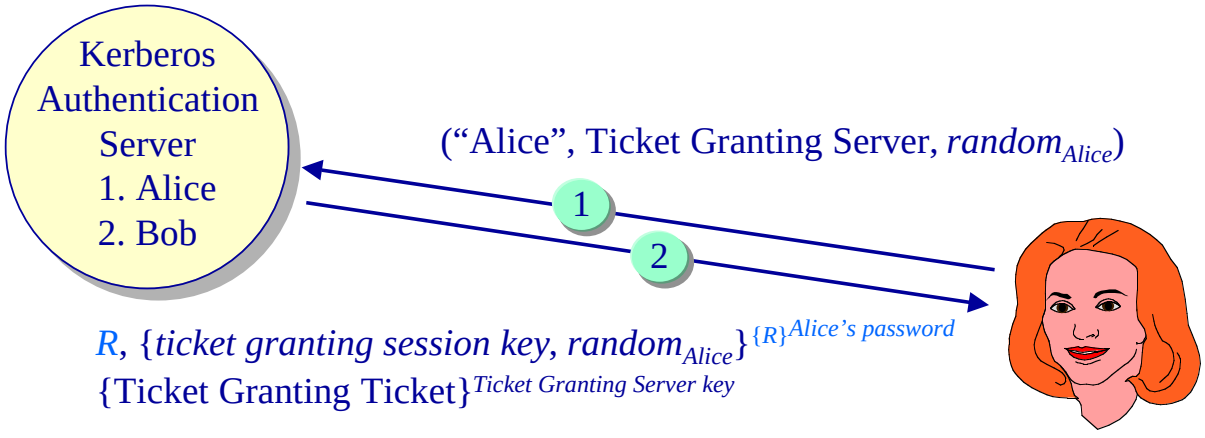
Using one-time passwords



- User uses a portable encryption device with a (large) key shared between the system and the device
  - » The system generates a random number and asks the user to encrypt it
  - » The user enters the number into her device and returns the result to the system as part of the log-in process

Avoiding Password Guessing/Spoofing Attacks

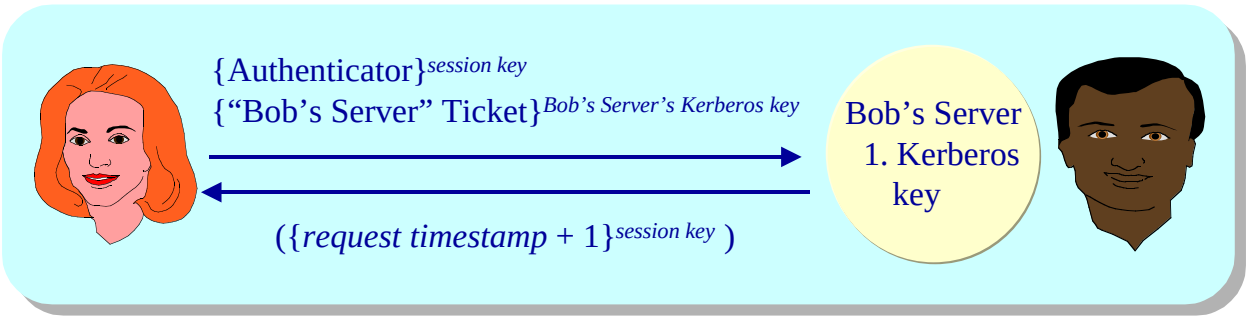
Using one-time passwords



- System generates a random number R and transmits to user
- User enters (traditional) password and someone computes $\{R\}^{password}$
- $\{R\}^{password}$ is used as the encryption key for the Kerberos server's response (the ticket granting session key)

Weaknesses of Kerberos

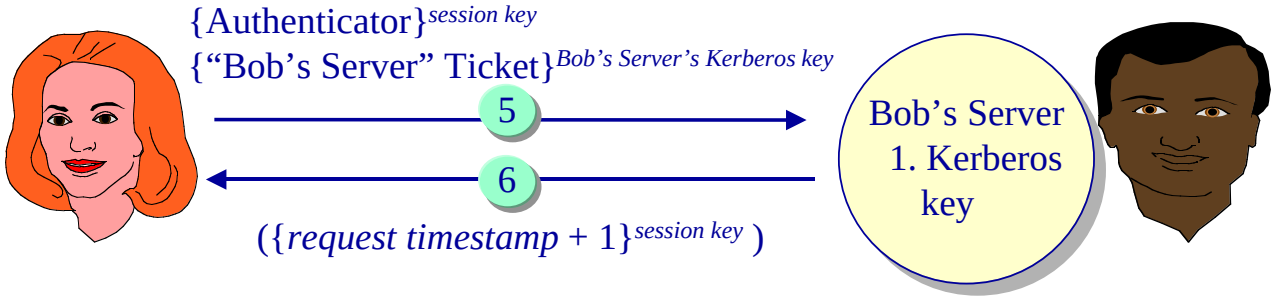
Security against replay attacks

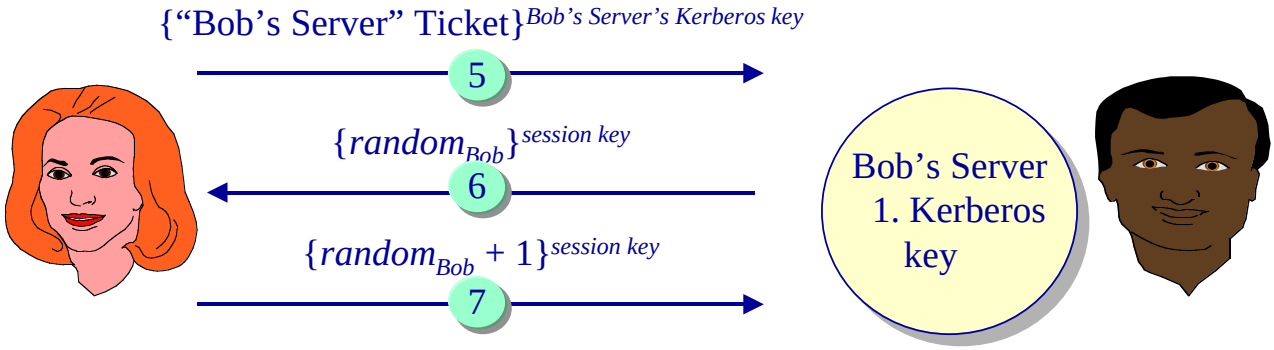


- Do timestamps effectively guard against replay attacks?
  - » Ideally one should not rely on tightly synchronized clocks
  - » Authenticators can be recorded and reused for 5 minutes
- Solution: Servers are allowed to cache past requests with still valid timestamps
  - » Caching authenticators problematic for servers using sockets or RPC
- Are time services secure?

Guarding Against Replay Attacks

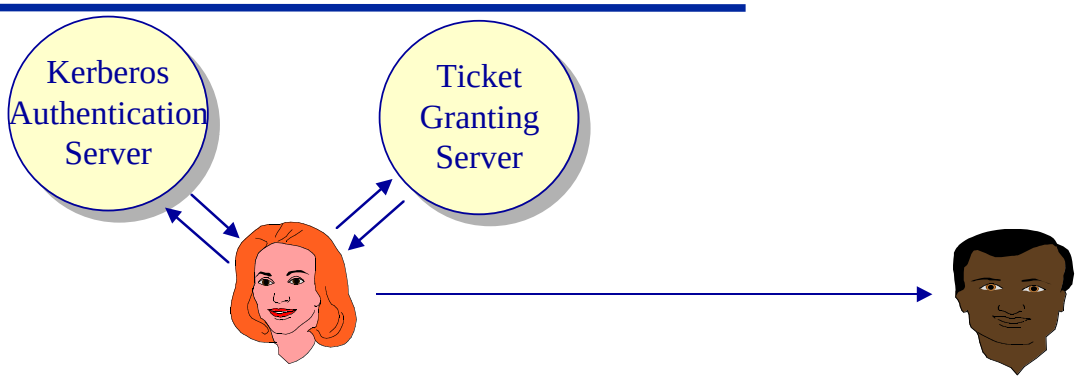
Replace authenticator with a server-initiated challenge





Authentication in Kerberos

Summary



- Kerberos — An implementation of the Needham & Schroeder protocol
  - » Encryption is based on DES
  - » Timestamps added to defend against replay attacks
- Works well if mutual trust exists between:
  - » clients
  - » servers
  - » the Kerberos authentication server
  - » a network time service