

NIS/CpE 691A

Information System Security

**Stevens Institute of Technology
Spring 2006**

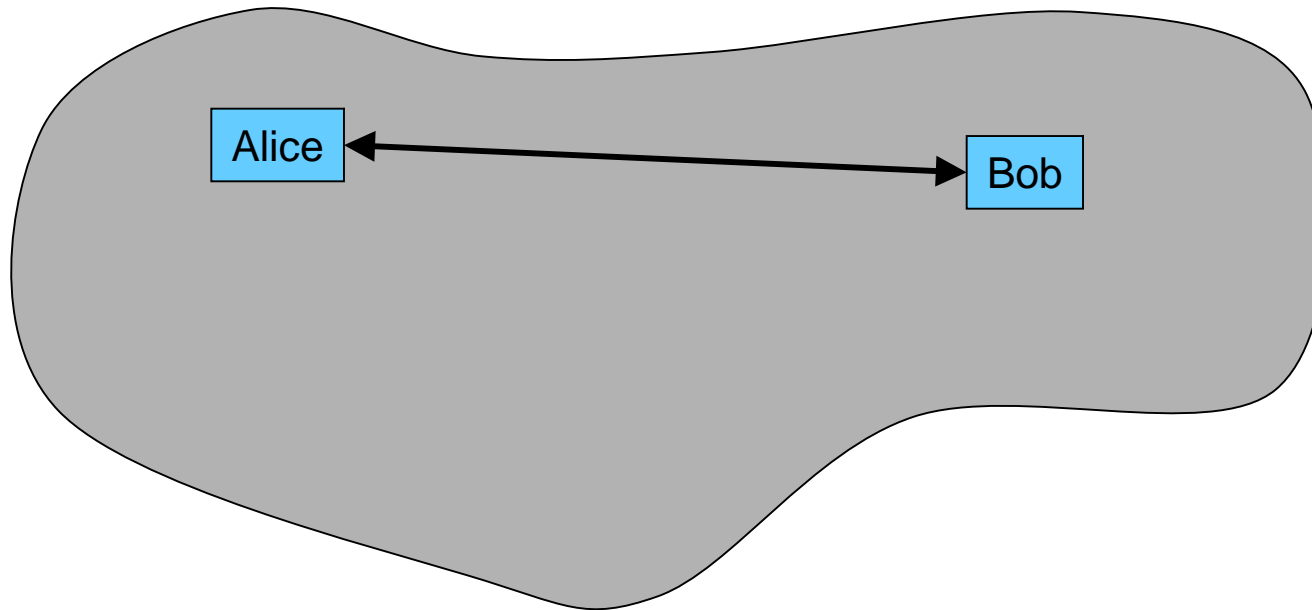
Class 5 – 2/16/06

Kerberos Authentication - Motivation



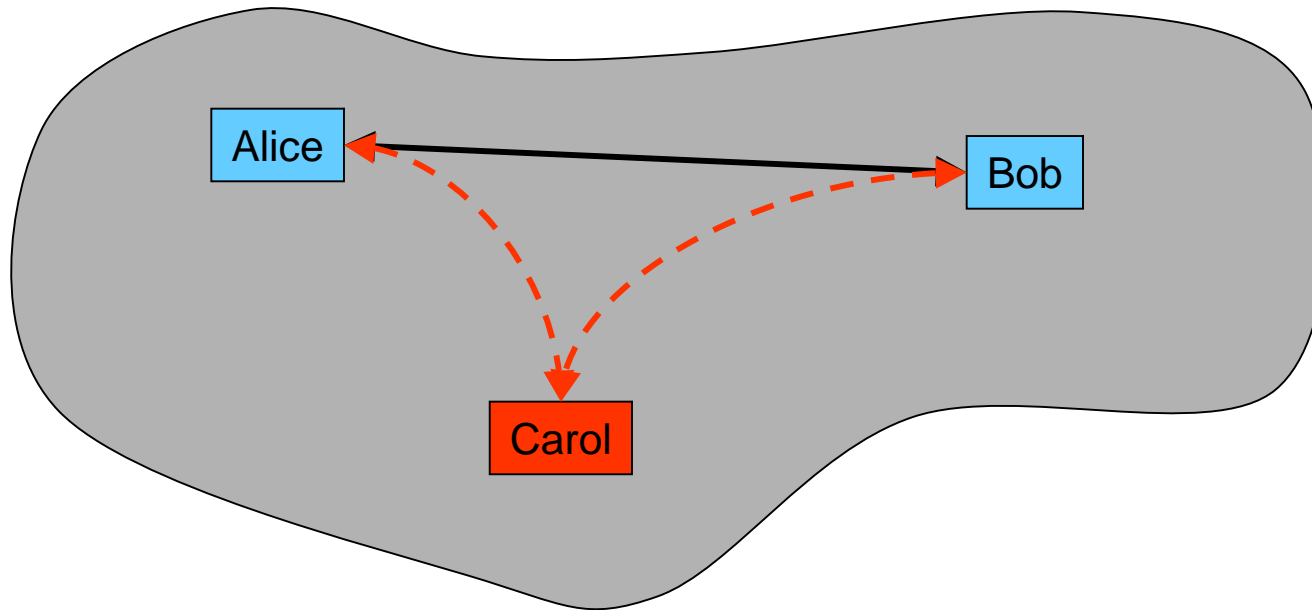
- Alice and Bob want to exchange information securely
- To do so, they need to authenticate each other

Kerberos Authentication - Motivation



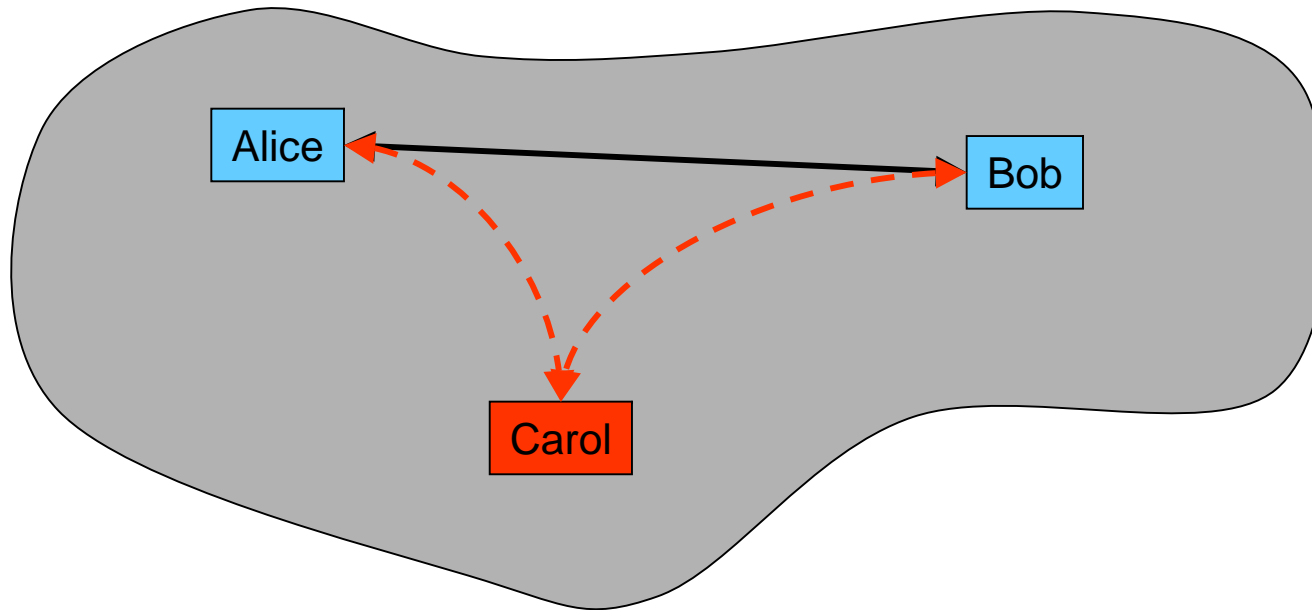
- Alice and Bob want to exchange information securely
- To do so, they need to authenticate each other
- They are operating in an insecure environment

Kerberos Authentication - Motivation



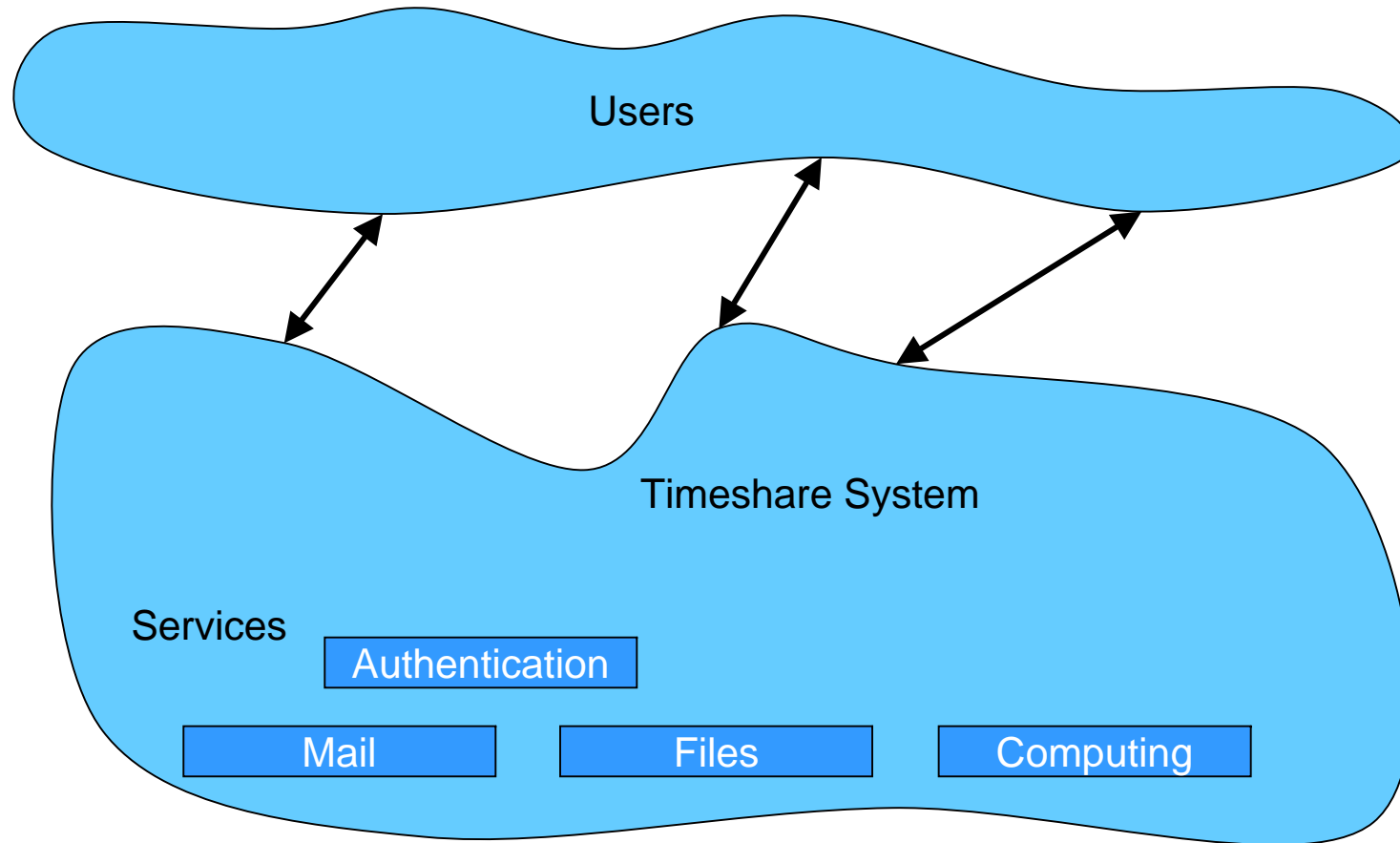
- Alice and Bob want to exchange information securely
- To do so, they need to authenticate each other
- They are operating in an insecure environment
- Carol is an interloper who can monitor all of the traffic between Alice and Bob
- Carol might be able to insert arbitrary messages into the stream

Kerberos Authentication - Motivation

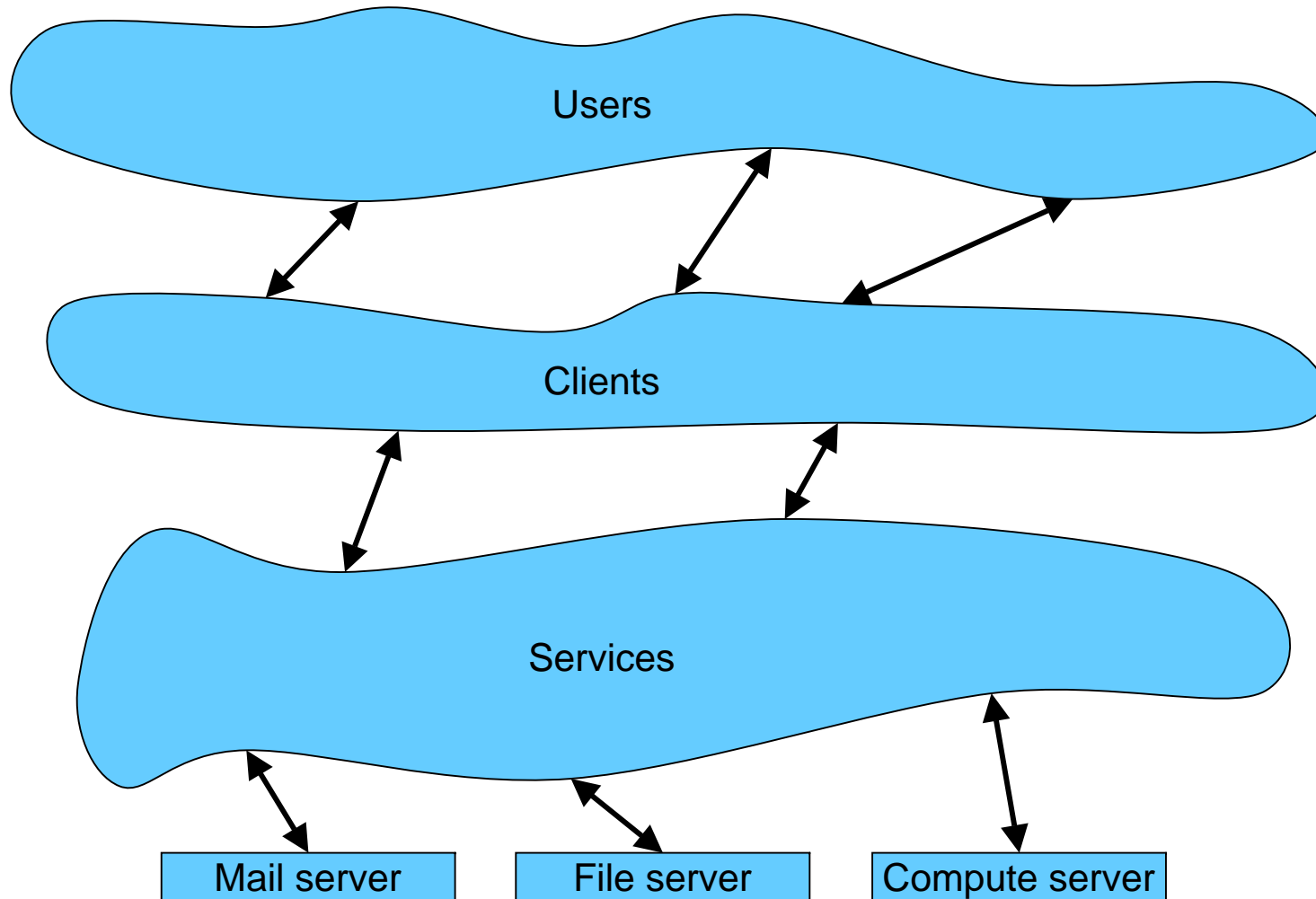


- Alice and Bob want to exchange information securely
- To do so, they need to authenticate each other
- They are operating in an insecure environment
- Carol is an interloper who can monitor all of the traffic between Alice and Bob
- Carol might be able to insert arbitrary messages into the stream
- **How do we provide a secure (authenticated) communications path?**

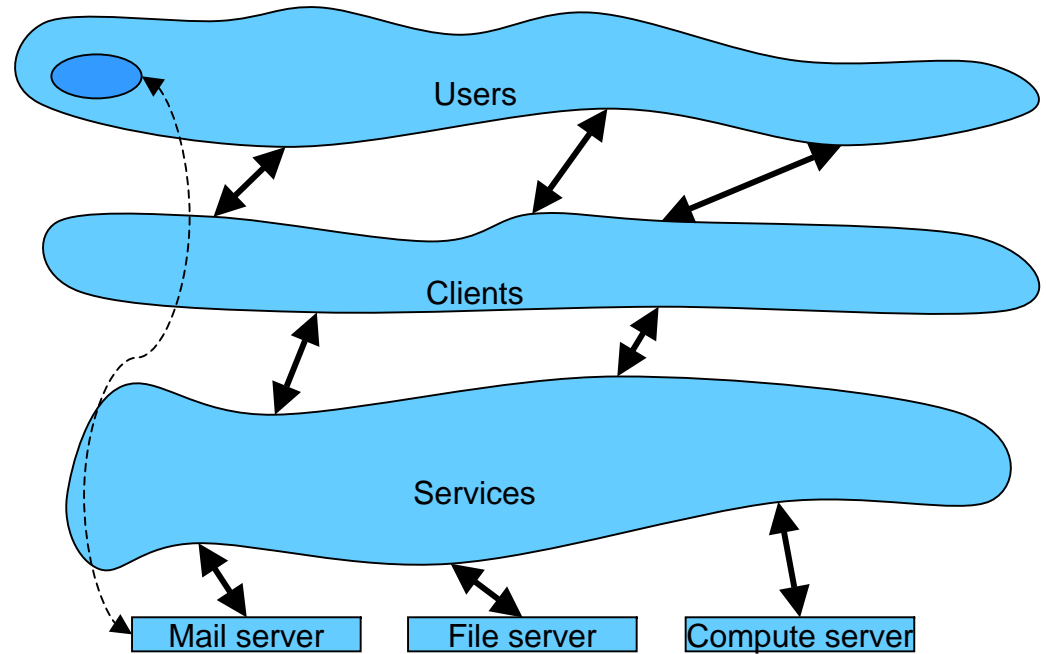
Authentication Before Distributed Computing



Distributed Computing Systems More than Alice, Bob and Carol

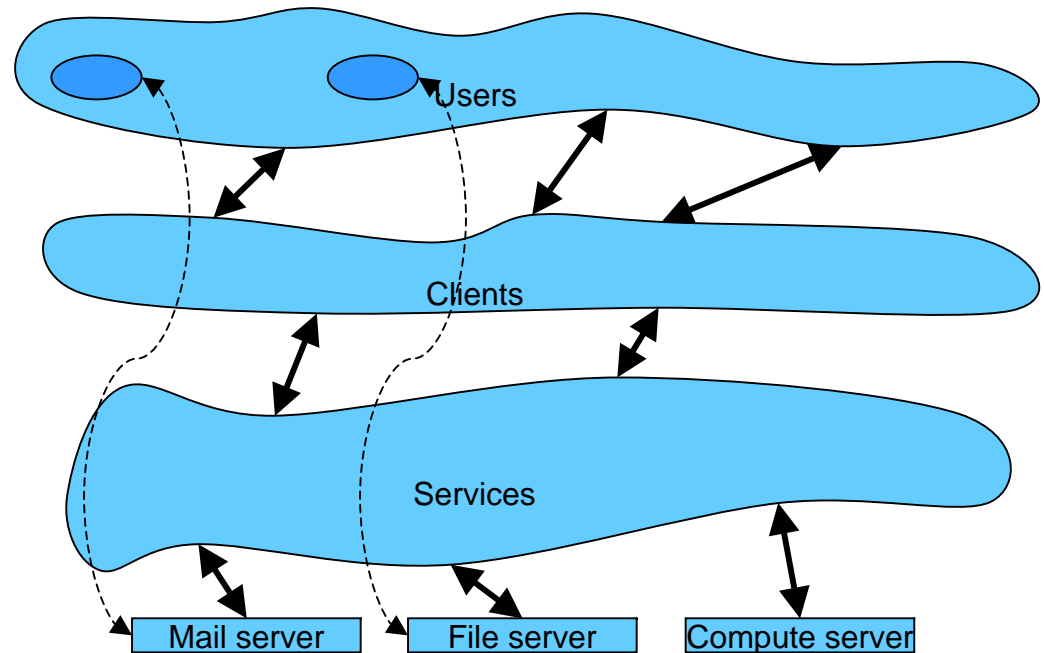


Kerberos - first cut



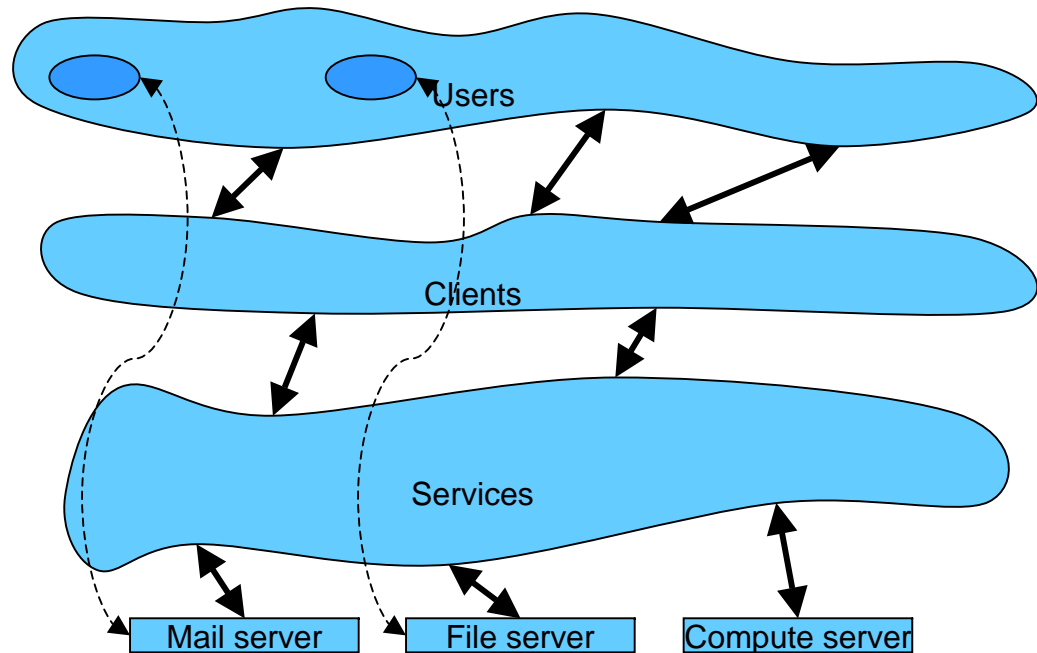
Distributed Authentication v0.1

- Each user is separately authenticated to each server



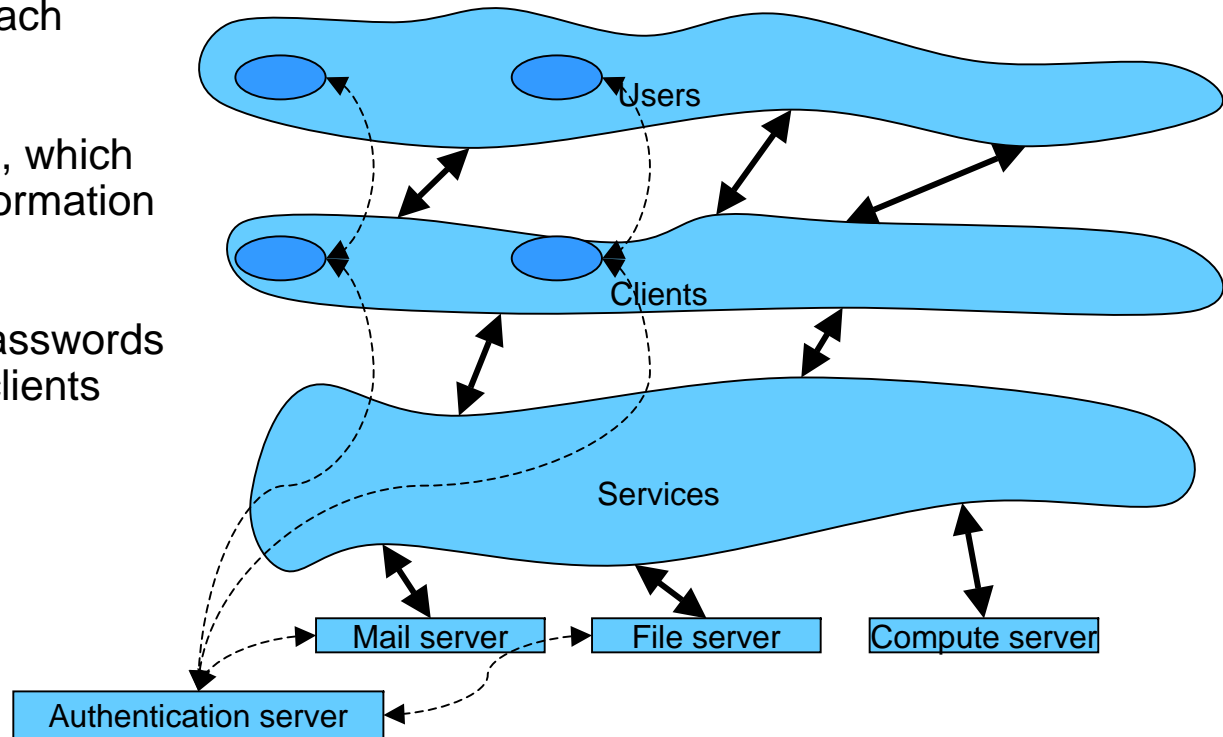
Distributed Authentication v0.1

- Each user is separately authenticated to each server
- For N users and M servers, requires $N \cdot M$ passwords
- A new user or user password change requires M server updates



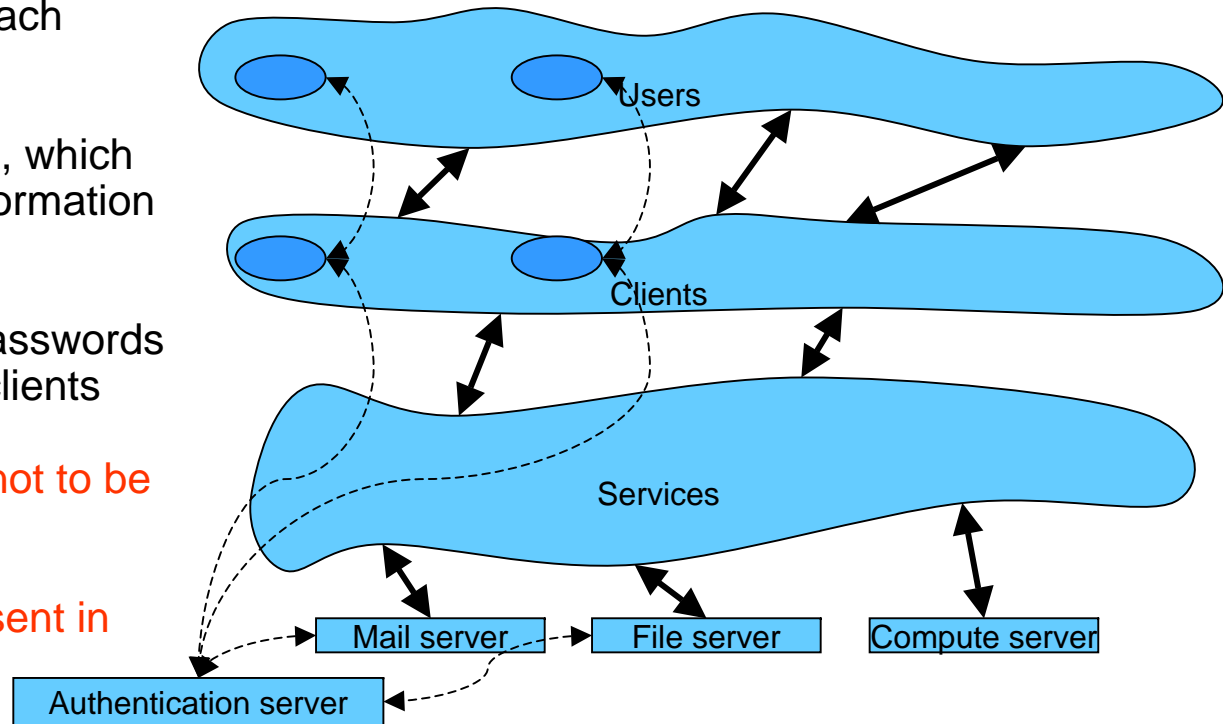
Distributed Authentication v0.2

- Use a centralized authentication server.
- It knows everyone's password and a password for each service
- Users log in to clients, which get authentication information from AS
- The AS has server passwords and passes them to clients



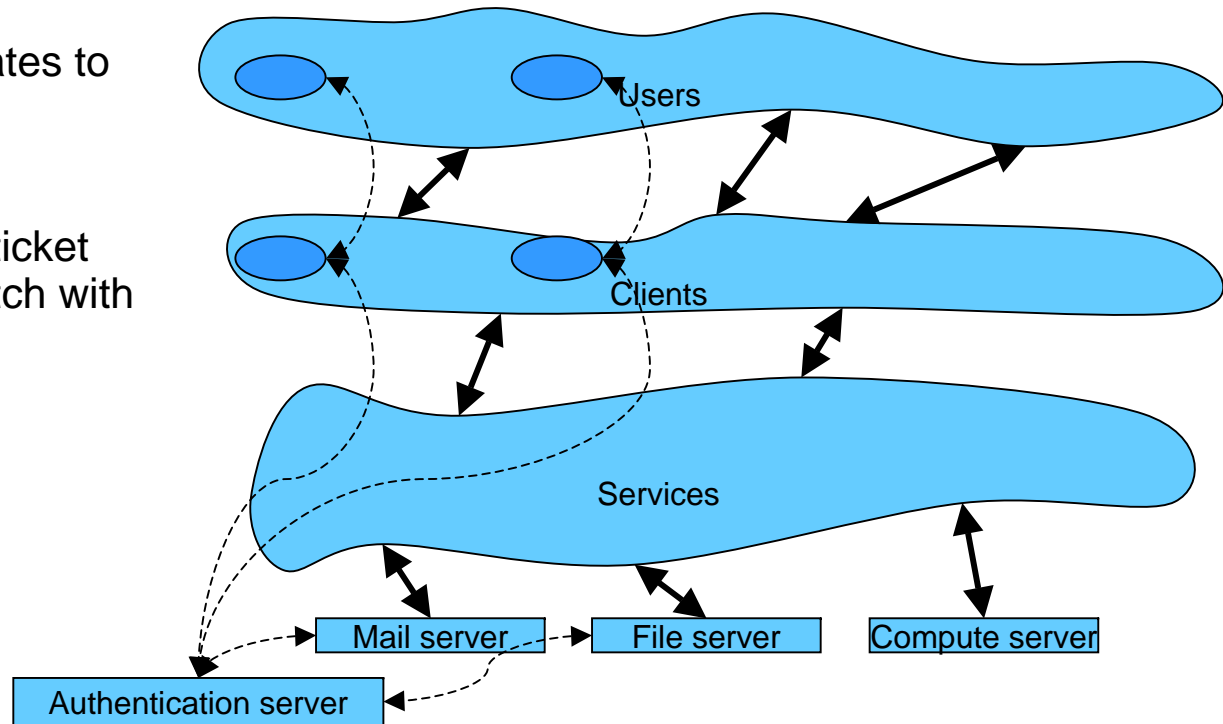
Distributed Authentication v0.2

- Use a centralized authentication server.
- It knows everyone's password and a password for each service
- Users log in to clients, which get authentication information from AS
- The AS has server passwords and passes them to clients
- Do you trust a client not to be compromised?
- Are user passwords sent in the clear?
- Are service passwords given to client?



Distributed Authentication v0.3

- AS generates “tickets”
- Mail server ticket:
 $TICKET = E_{mail}\{Alice\}$
- User/client authenticates to mail server with
 $\{Alice, TICKET\}$
- Mail server decrypts ticket with its key, finds match with “Alice”



Distributed Authentication v0.3

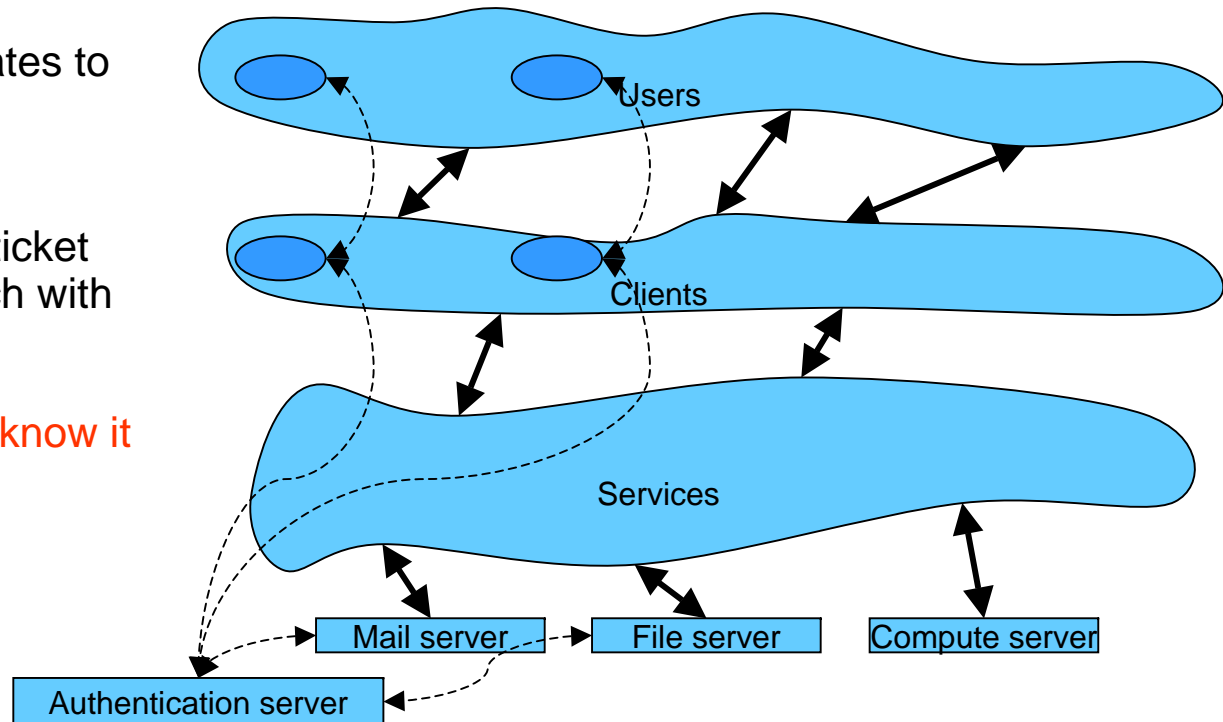
- AS generates “tickets”

- Mail server ticket:
 $TICKET = E_{mail}\{Alice\}$

- User/client authenticates to mail server with
 $\{Alice, TICKET\}$

- Mail server decrypts ticket with its key, find match with
“Alice”

- How does the server know it decrypted a valid
userID=Alice?



Distributed Authentication v0.4

- AS generates “tickets”

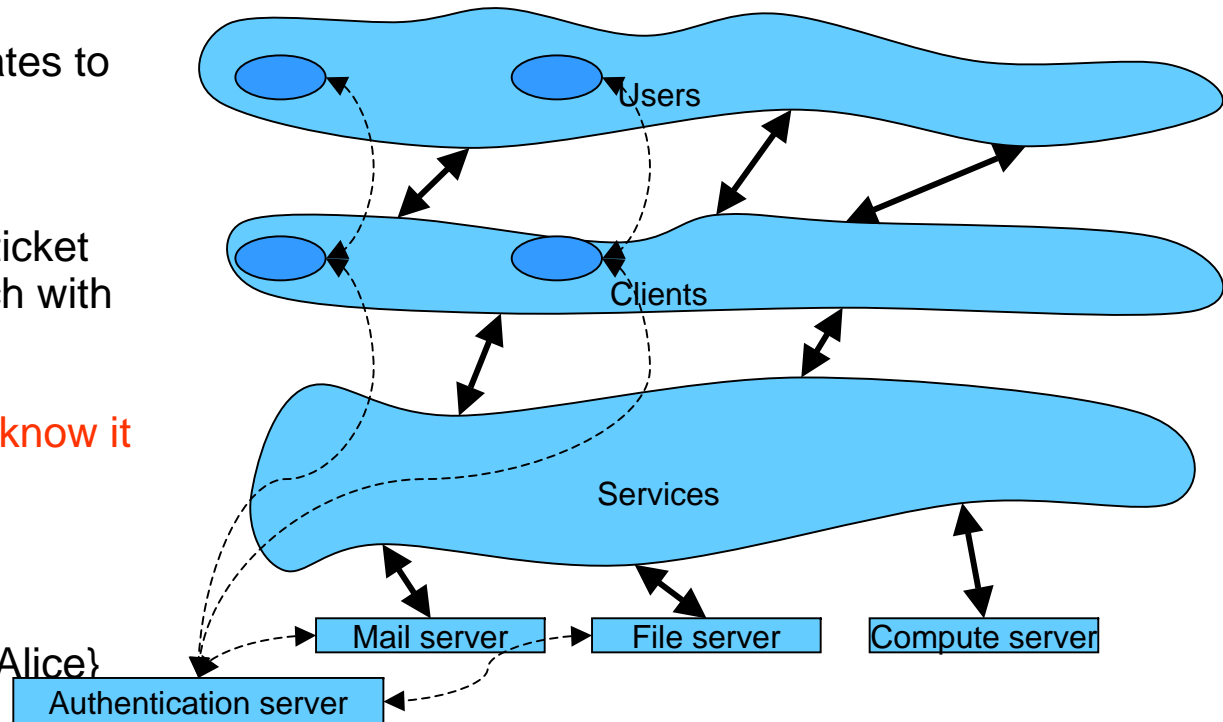
- Mail server ticket:
 $TICKET = E_{mail}\{Alice\}$

- User/client authenticates to mail server with
 $\{Alice, TICKET\}$

- Mail server decrypts ticket with its key, find match with “Alice”

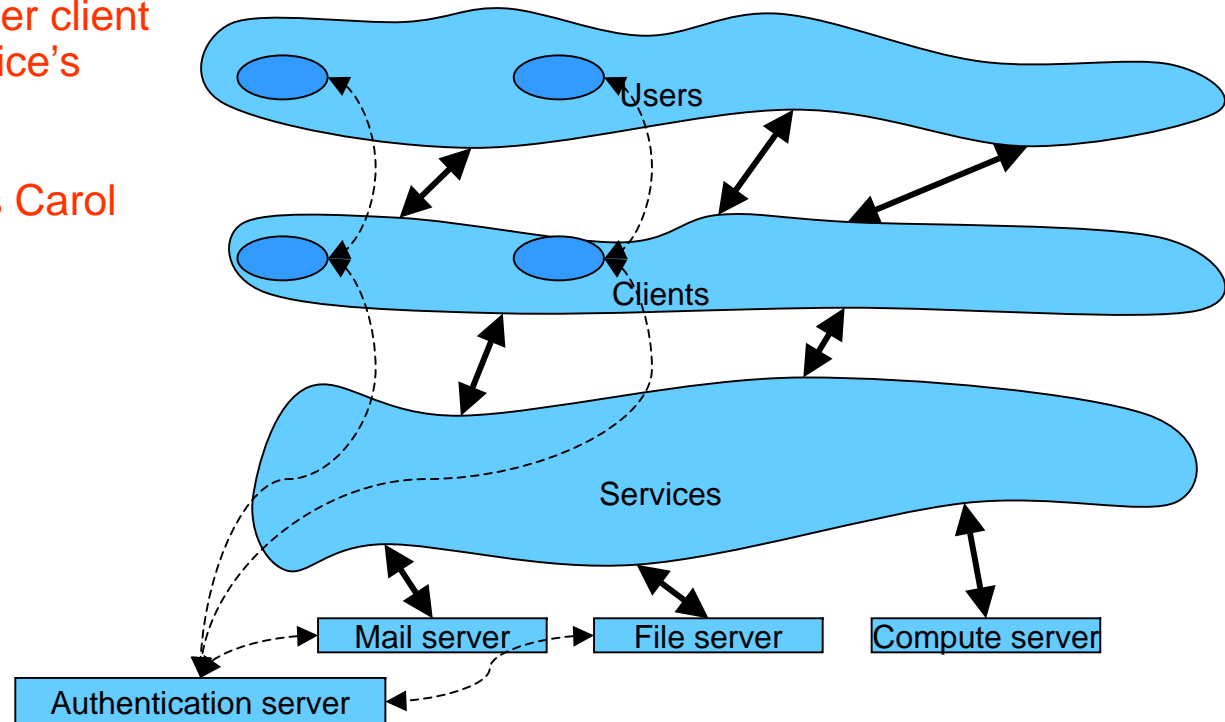
- How does the server know it decrypted a valid $userID=Alice$?

- Make the ticket:
 $TICKET = E_{mail}\{mail, Alice\}$



Distributed Authentication v0.4

- $TICKET = E_{mail}\{mail, Alice\}$
- What about replay attacks?
- Carol compromises her client to copy and replay Alice's ticket
- The mail server gives Carol Alice's mail.



Distributed Authentication v0.5

- $TICKET = E_{mail}\{mail, Alice\}$

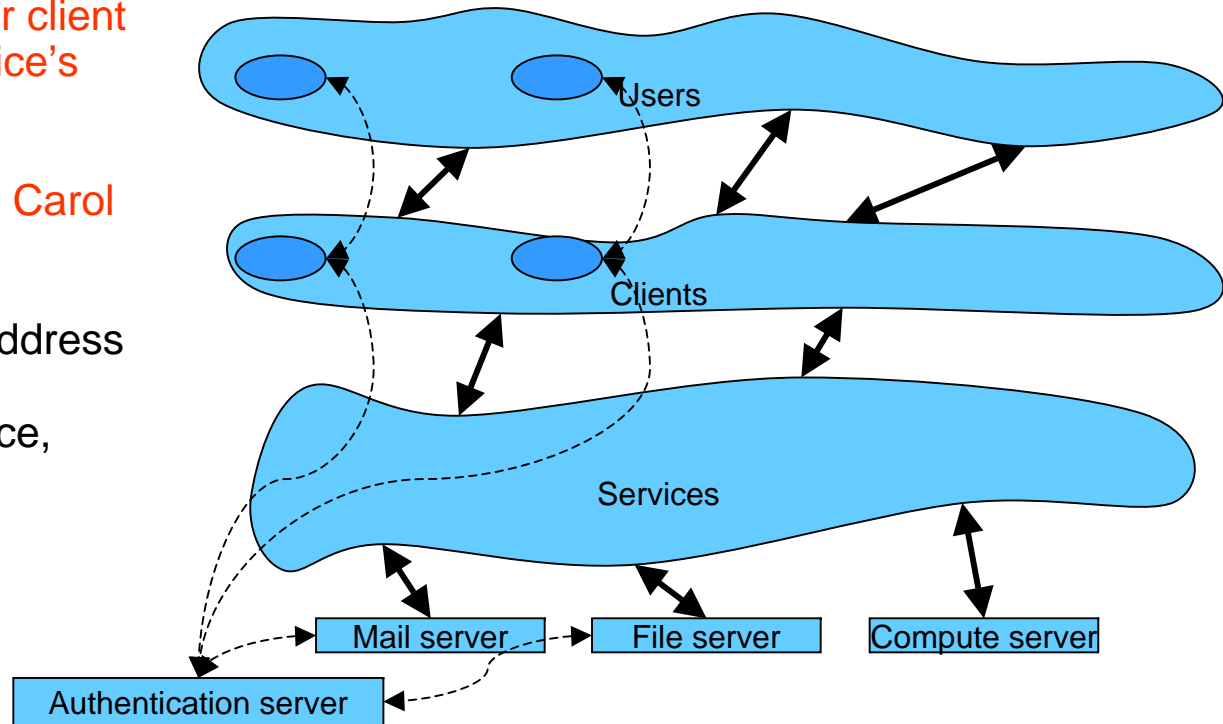
- What about replay attacks?

- Carol compromise her client to copy and replay Alice's ticket

- The mail server gives Carol Alice's mail.

- A fix: include client address

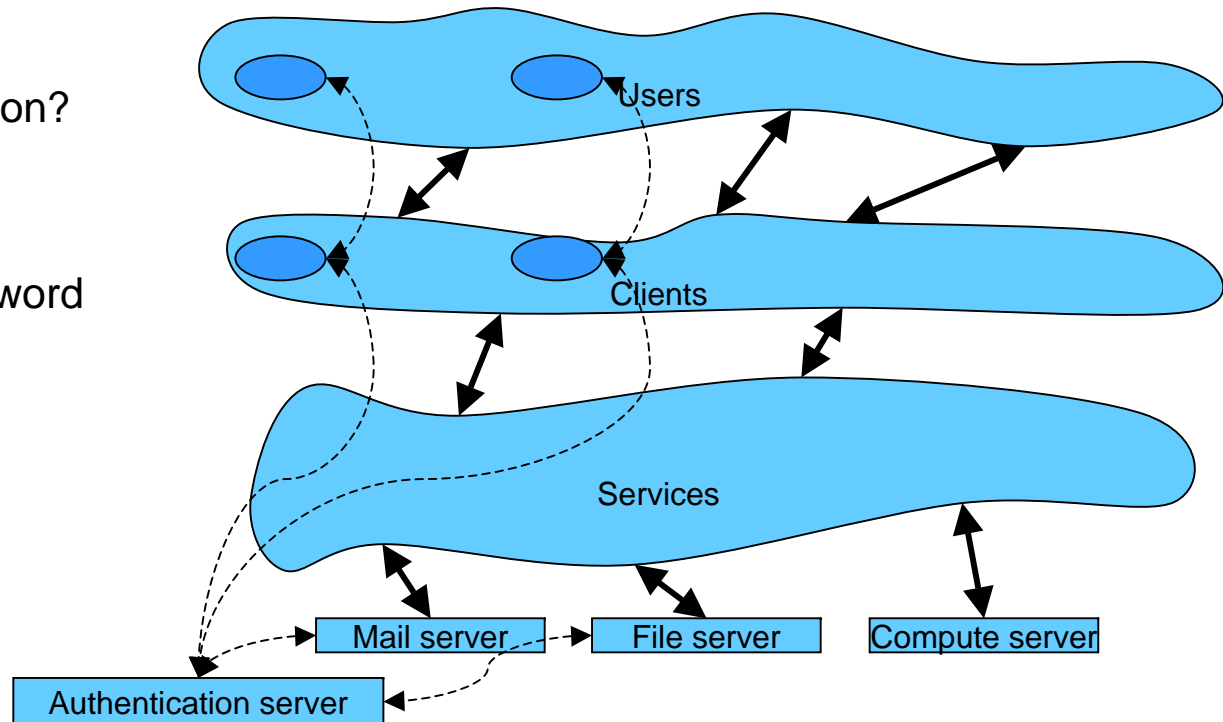
$TICKET = E_{mail}\{mail, Alice, Alice_client_addr\}$



Distributed Authentication v0.5

$TICKET = E_{mail} \{mail, Alice, Alice_client_addr\}$

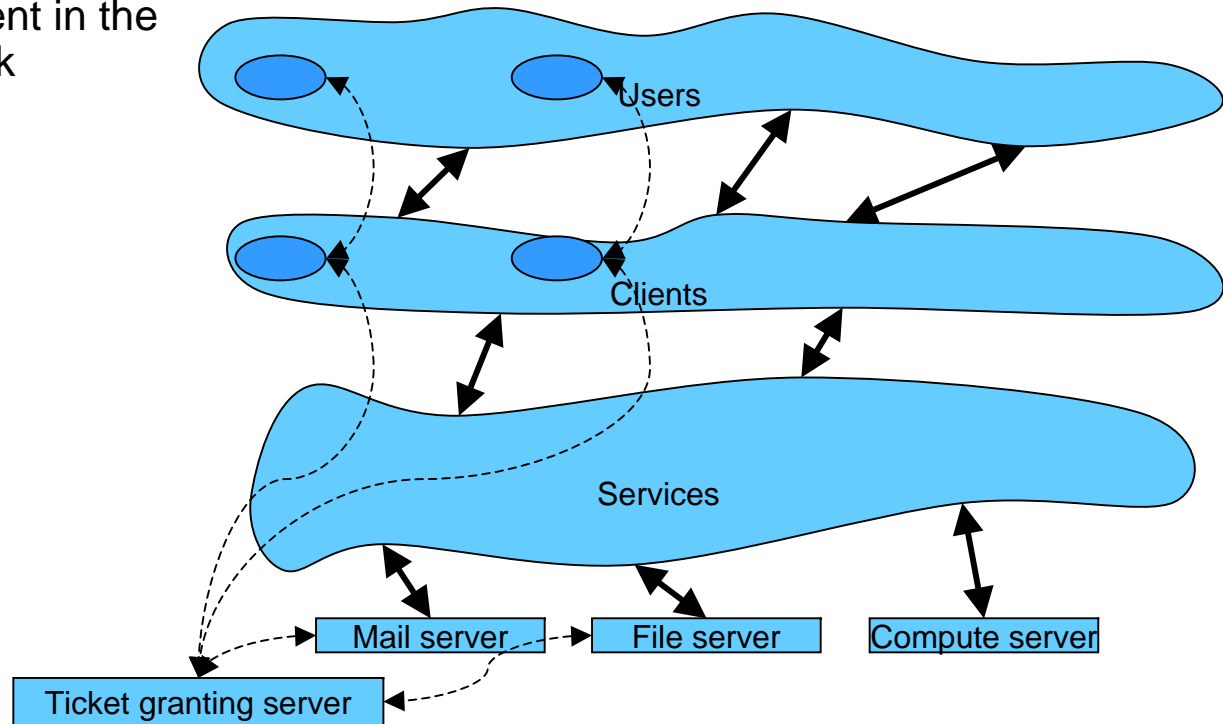
- What is the lifetime of a ticket?
 - one time?
 - one terminal session?
 - one day?
 - forever?
- How is the user password sent to the AS?
 - in the clear?
 - encrypted? How?



Distributed Authentication v0.5.1

Security requirements:

- Users enter their passwords once at beginning of client session
- Passwords are not sent in the clear over the network



Distributed Authentication v0.5.1

Security requirements:

- Users enter their passwords once at beginning of client session
- Passwords are not sent in the clear over the network

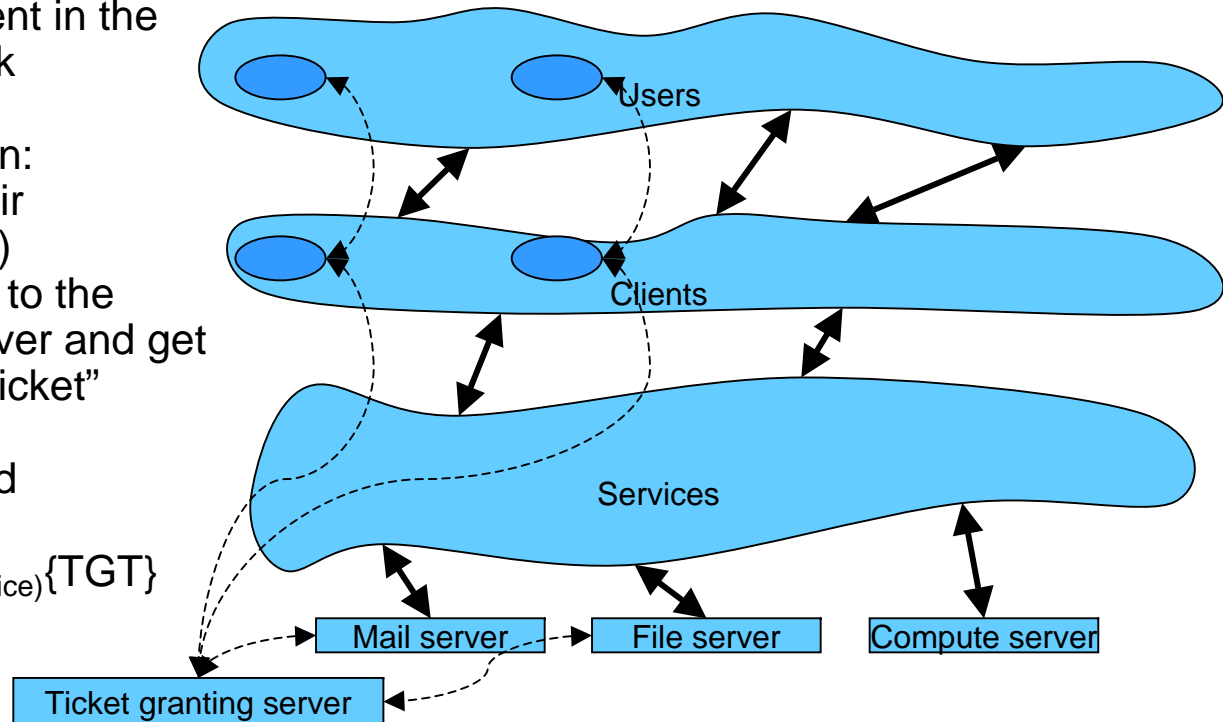
- Authentication session:
 - Users log in to their workstation (client)
 - They authenticate to the ticket granting server and get a “ticket-granting ticket”

User->Client: password

Client->TGS: Alice

TGS->Client: $E_{\text{passwd}(\text{Alice})}\{\text{TGT}\}$

Client decrypts TGT



Distributed Authentication v0.5.1

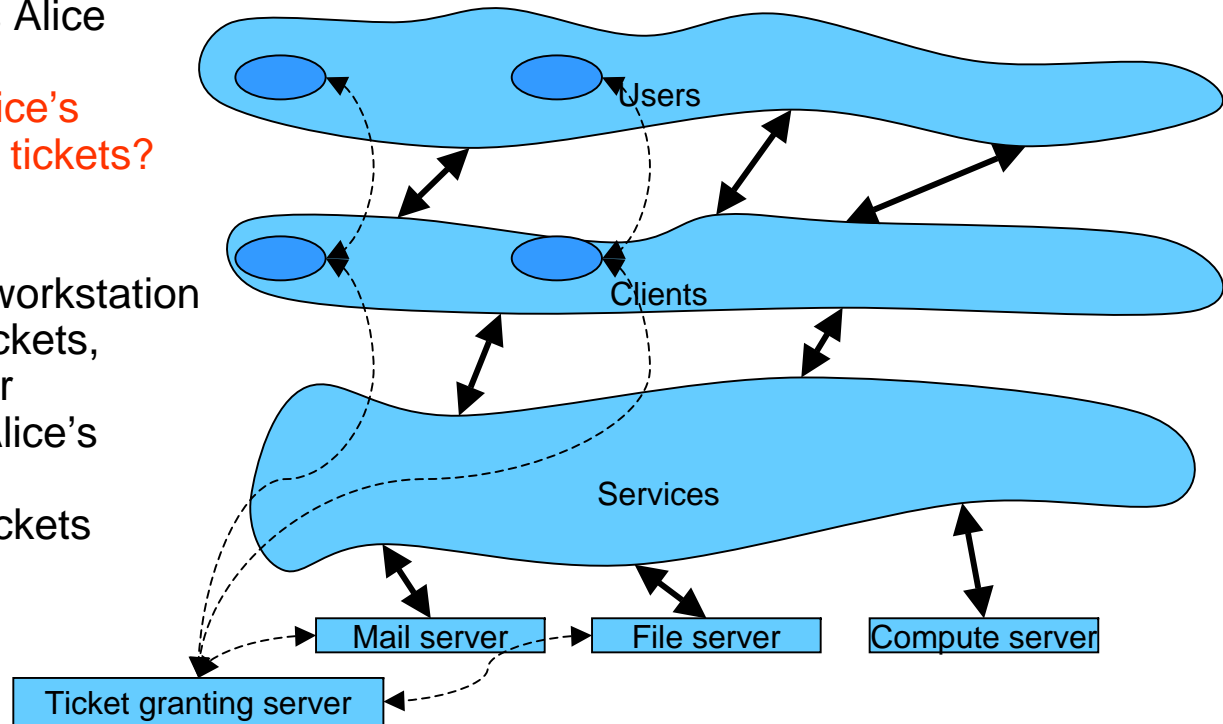
How reusable are tickets?

Alice finishes session, logs out
Carol logs in, finds left-over tickets
Carol masquerades as Alice

Can you depend on Alice's
processes destroying tickets?

Even if you do:

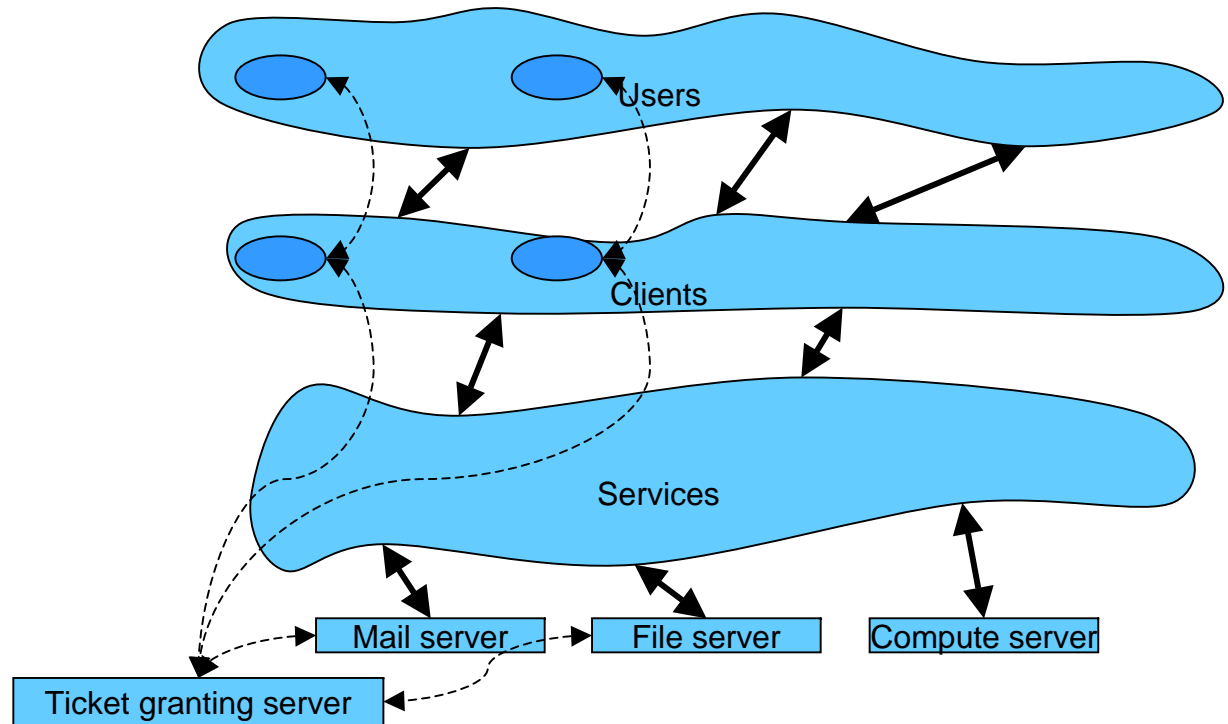
Carol's compromised workstation
copies all of Alice's tickets,
Carol later modifies her
workstation to send Alice's
workstation address
Carol replays Alice's tickets



Distributed Authentication v0.5.2

Tickets must have a lifetime.

$TICKET = E_{mail} \{mail, Alice, Alice_addr, expiration, timestamp\}$



Distributed Authentication v0.5.2

Tickets must have a lifetime.

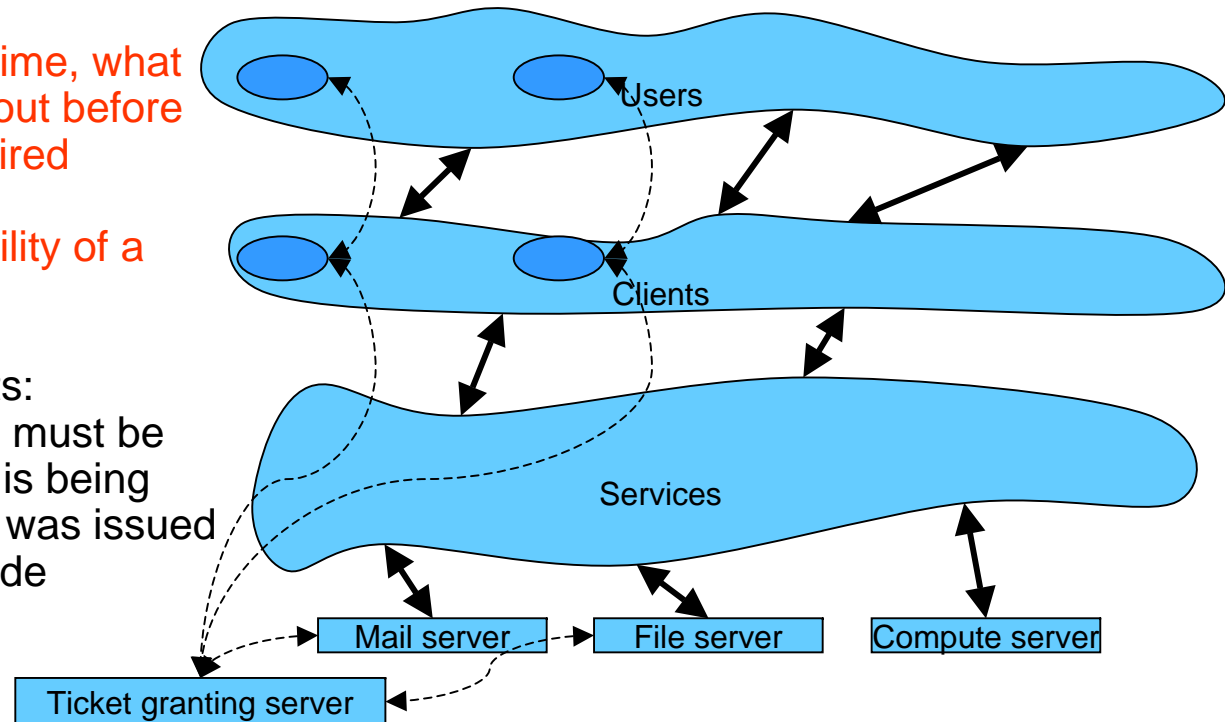
$TICKET = E_{mail} \{mail, Alice, Alice_addr, expiration, timestamp\}$

But if it is a **FIXED** lifetime, what about users who log out before their tickets have expired

What about the possibility of a bogus TGS?

Additional requirements:

- Network service (NS) must be able to prove a ticket is being used by the person it was issued to [tickets must encode

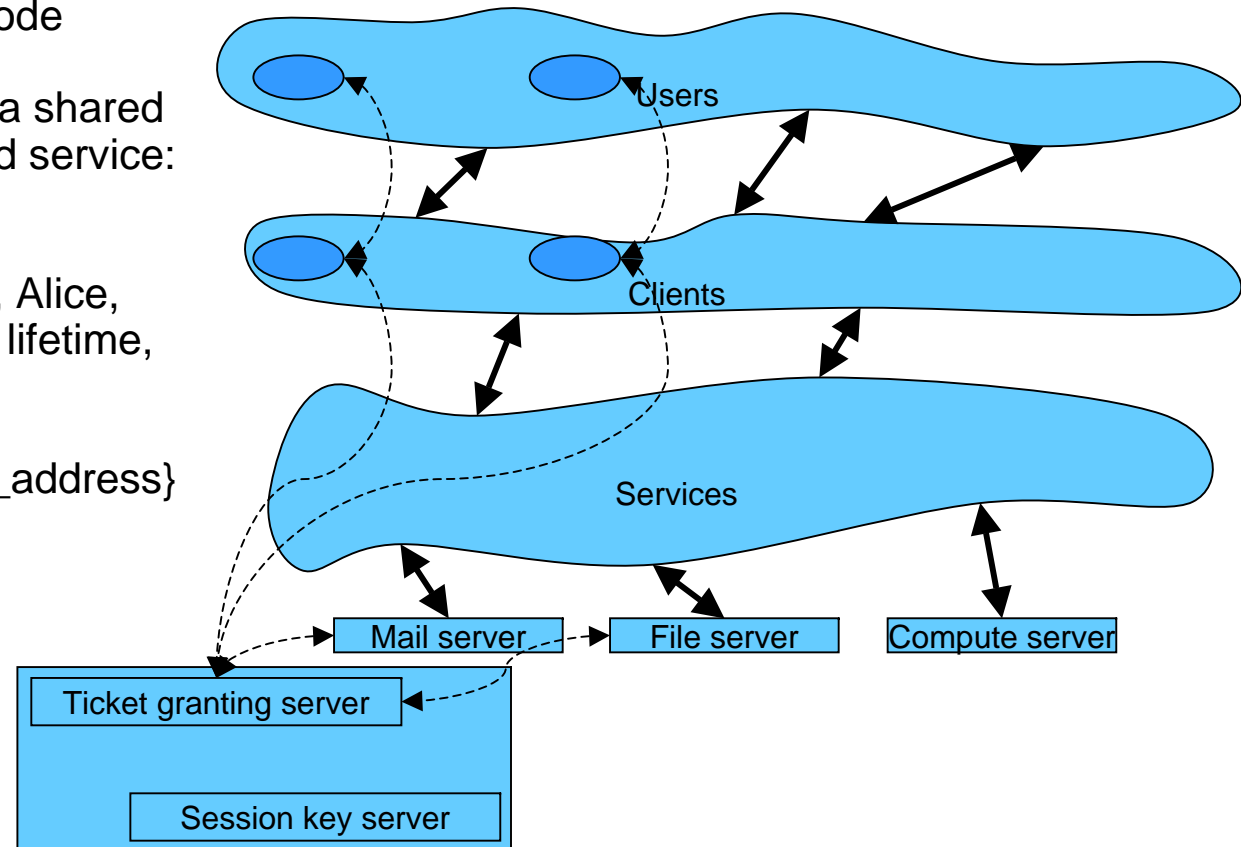


Distributed Authentication v0.5.3

Additional requirements:

- Network service (NS) must be able to prove a ticket is being used by the person it was issued to [tickets must encode to]
- Encrypt the ticket in a shared key with the user and service:

[sessionkey, ticket]
TICKET={sessionkey, Alice,
Alice_address, mail, lifetime,
timestamp}
AUTHENTICATOR=
 $E_{\text{sessionkey}}\{\text{Alice, Alice_address}\}$



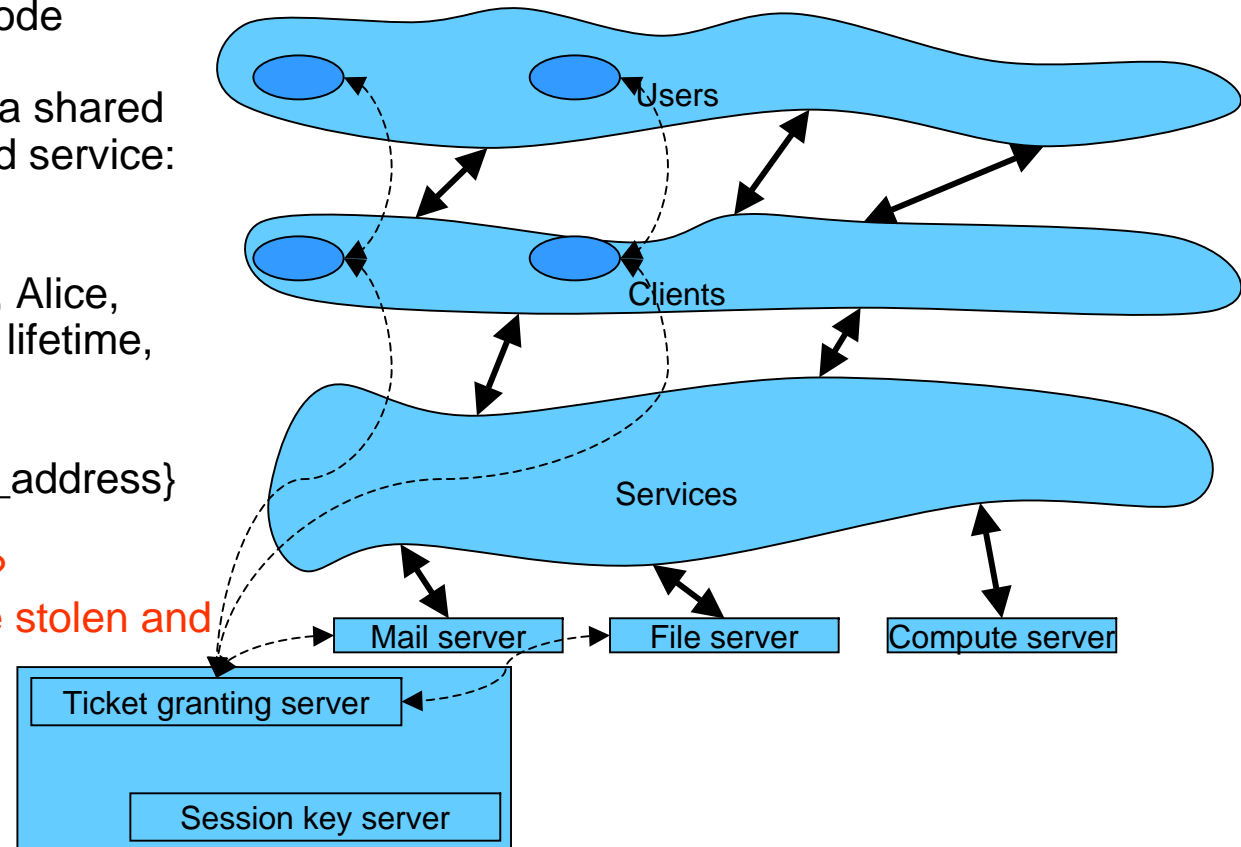
Distributed Authentication v0.5.3

Additional requirements:

- Network service (NS) must be able to prove a ticket is being used by the person it was issued to [tickets must encode to]
- Encrypt the ticket in a shared key with the user and service:

[sessionkey, ticket]
TICKET={sessionkey, Alice,
Alice_address, mail, lifetime,
timestamp}
AUTHENTICATOR=
 $E_{\text{sessionkey}}\{\text{Alice, Alice_address}\}$

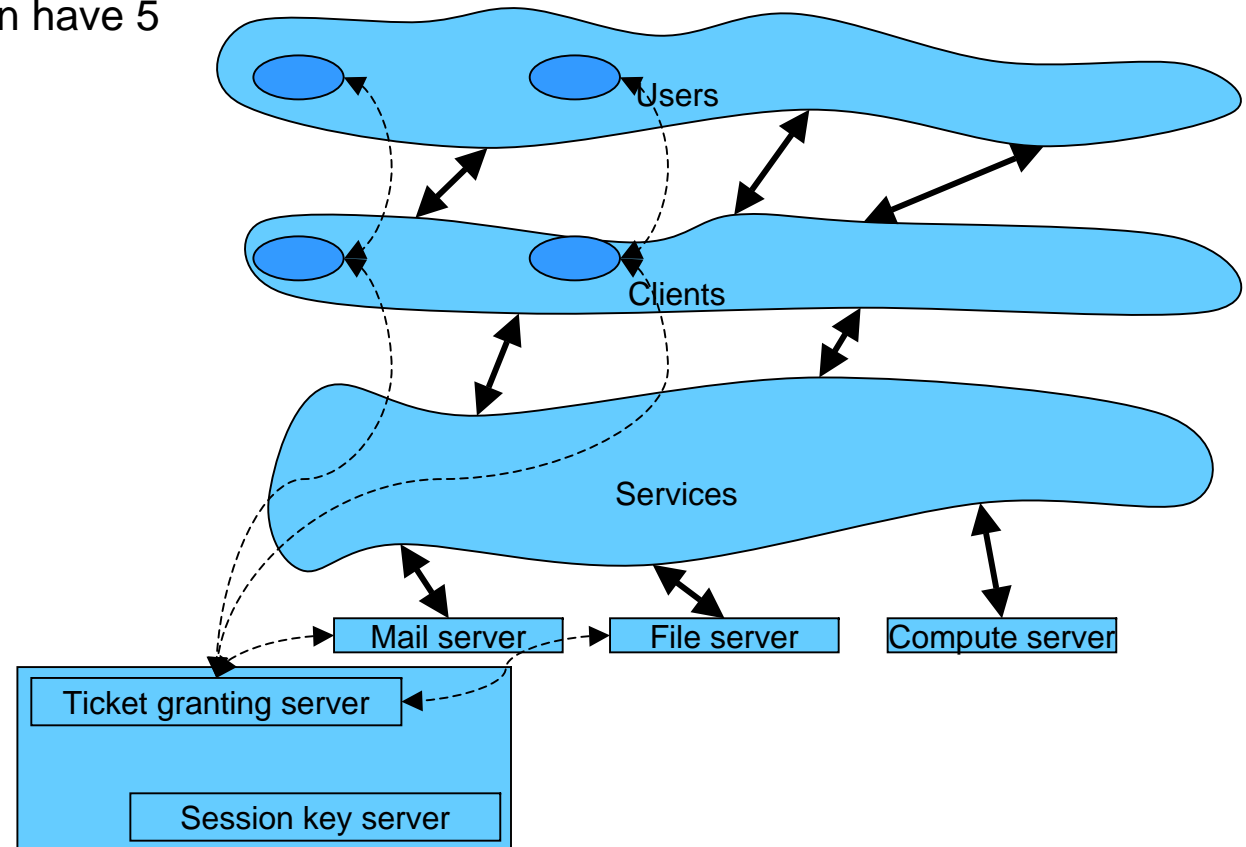
What does this solve?
Authenticators can be stolen and
replayed.



Distributed Authentication v0.5.3

What does this solve?
Authenticators can be stolen and
replayed.

But Authenticators can have 5
minute lifetimes...

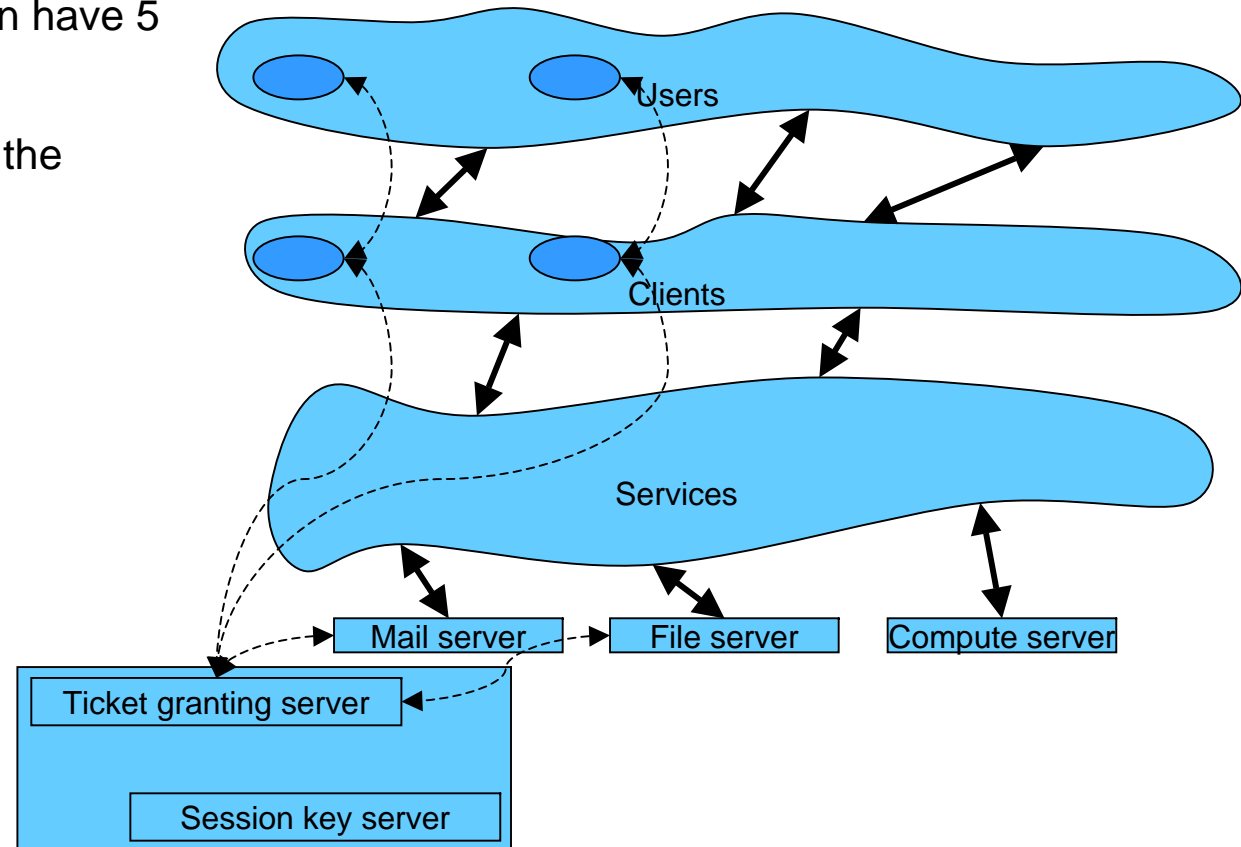


Distributed Authentication v0.5.3

What does this solve?
Authenticators can be stolen and
replayed.

But Authenticators can have 5
minute lifetimes...

So does this solve all the
problems?



Distributed Authentication v0.5.3

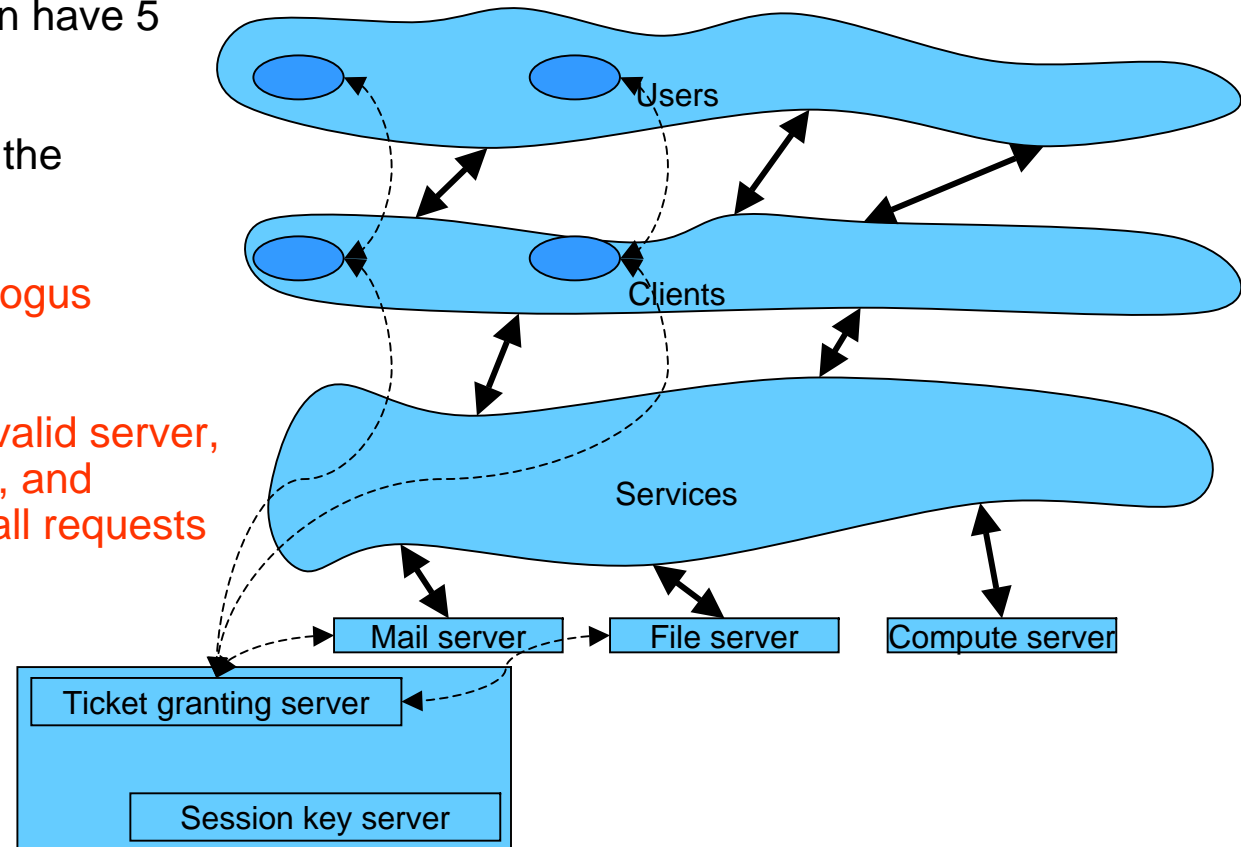
What does this solve?
Authenticators can be stolen and replayed.

But Authenticators can have 5 minute lifetimes...

So does this solve all the problems?

No - what about the bogus server?

It masquerades as a valid server, doesn't verify tickets, and appears to process all requests



Distributed Authentication v0.5.4

What does this solve?
Authenticators can be stolen and replayed.

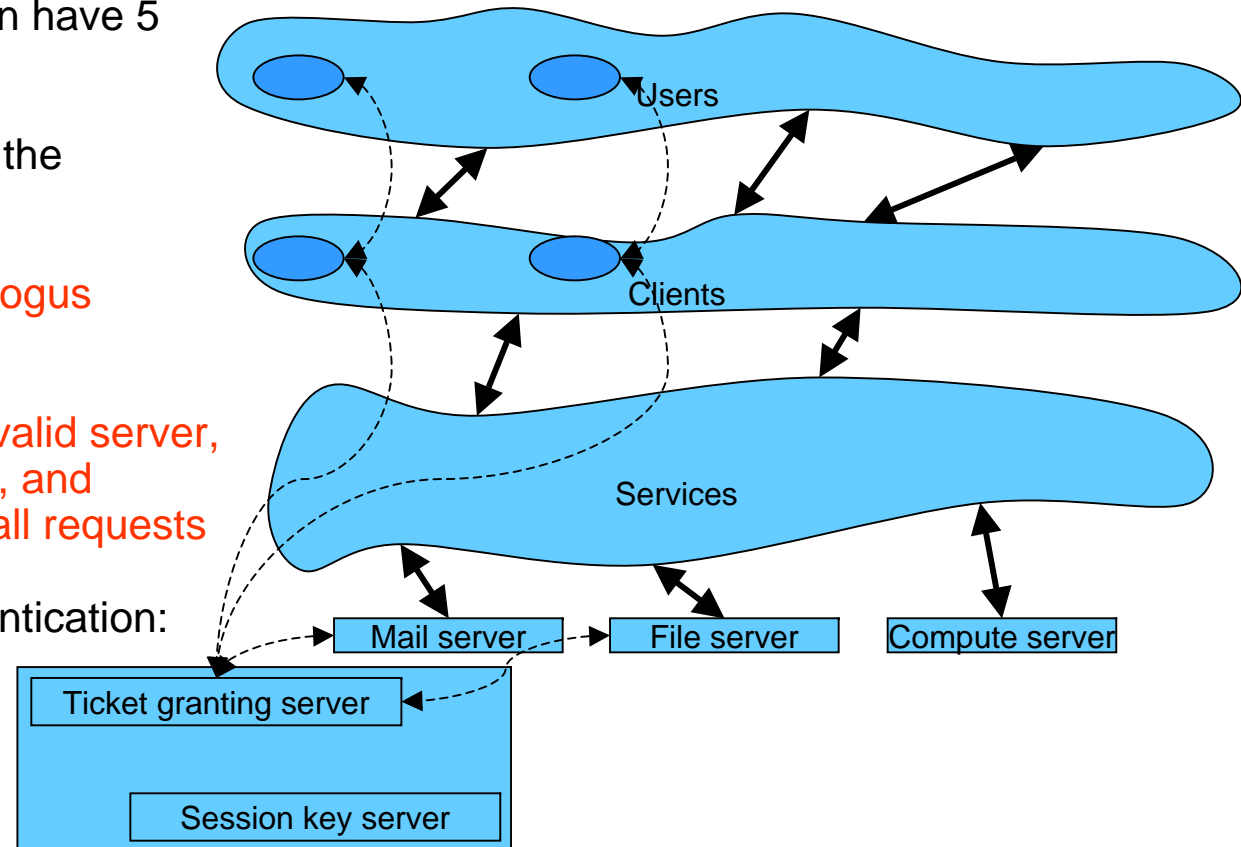
But Authenticators can have 5 minute lifetimes...

So does this solve all the problems?

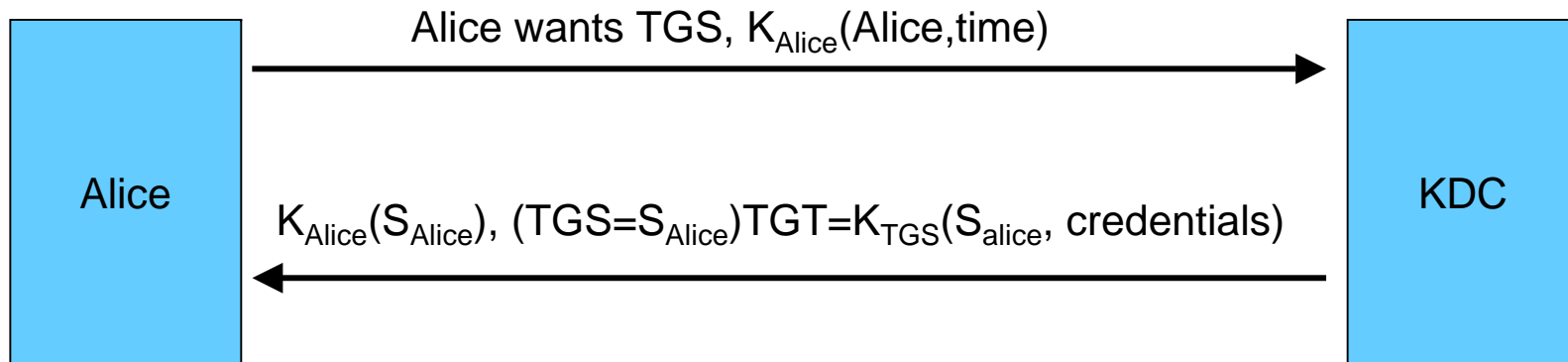
No - what about the bogus server?

It masquerades as a valid server, doesn't verify tickets, and appears to process all requests

Require mutual authentication:
Alice -> mail,
mail -> Alice
before information
is sent

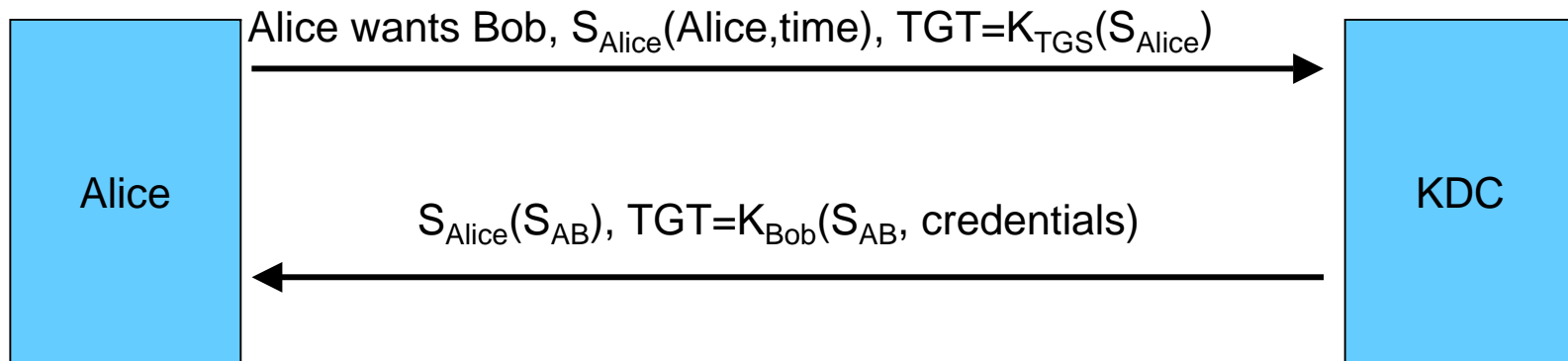


Kerberos V5 = Distributed Authentication V0.5.4



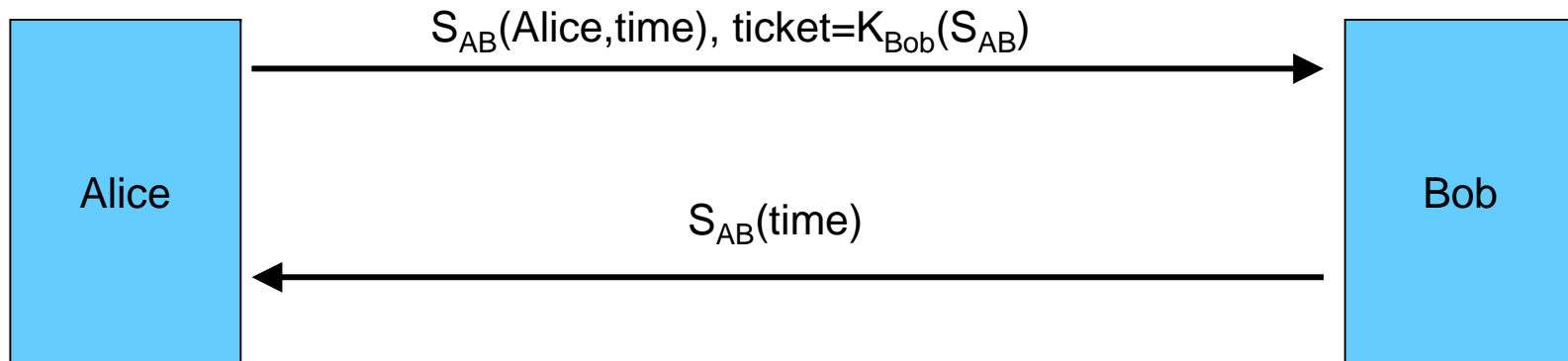
- AS (Authentication Service) exchange:
 - Alice types logon and password to client
 - Workstation converts password to encryption key and saves it
 - Client sends a Kerberos authentication request
 - KDC looks up Alice in database, verifies that it is Alice who has encrypted the request and the request is timely
 - KDC generates a session key, and encrypts in Alice's key
 - KDC generates a TGT, encrypting the credentials in the session key
 - Alice decrypts the session key and TGT

Kerberos V5 = Distributed Authentication V0.5.4



- TGS (Ticket Granting Service) exchange:
 - Alice's client requests credentials to use with Bob
 - Request includes Alice's name, an authenticator encrypted with session key, TGT, and name of service requested
 - KDC extracts Alice's session key and evaluates authenticator
 - KDC generates Alice-Bob key, encrypts with Bob's key and with Alice's session key
 - Alice's client decrypts the mutual session key to use with Bob

Kerberos V5 = Distributed Authentication V0.5.4



- CS (Client Server) exchange:

- Alice's sends authenticator to Bob, encrypted in mutual session key, along with mutual session key, encrypted in Bob's master key
- Bob decrypts session key and uses it to verify Alice's authenticator
- Bob uses current time, encrypted in mutual session key to authenticate to Alice
- Alice decrypts Bob's authenticator and verifies timeliness
- Session proceeds.

Kerberos V5 vs Kerberos V4

- With a fast workstation, 5 minute exposure of authenticator can be excessive. All authenticators are one-time. Which doesn't put a burden on fast workstations...
- Kerberos V4 doubly encrypted exchanges in TGS key as well as user's password. TGS encryption is sufficient and doesn't unnecessarily expose user password (as a master key)
- Similar change for TGS protocol: no need to encrypt in TGT key if exchange is encrypted in service's key.
- Ticket forwarding to allow operation in other domains
- Delegation of user access rights to a server
- Kerberos V4 was based on DES. Kerberos V5 adds options for other cryptographic algorithms (MS has extended to include public key system)

Class Project

- Research one of the topics listed below (individually or in group of 2-3). Topic selection, group members, and 1 paragraph paper abstract due by class 6 – 2/23/06.
- Prepare a paper reporting on the topic (6-10 pages) (due by class 14).
- Prepare a presentation on the topic to be presented to the class.
 - (20 minutes, including 5 minutes for questions, discussion) (PowerPoint 2000 slides due by class 13, presented in classes 13 & 14).
- Sample topics:
 - Security in Microsoft Vista® (capabilities or vulnerabilities)
 - Security techniques in Microsoft .Net Passport®
 - Computer virus detection techniques
 - State-of-the-art in Intrusion Detection Systems
 - AES cryptographic attacks
 - Attacks on SHA-1
 - Risk assessment tools
 - Biometric techniques
 - Internet worms/virii
 - Steganography
 - DMCA - issues in copy protection

 - I'm open to other topics - email/call/see me to discuss