Kerberos

CS60002: Distributed Systems

Antonio Bruto da Costa
Ph.D. Student, Formal Methods Lab,
Dept. of Computer Sc. & Engg.,
Indian Institute of Technology Kharagpur

Origins

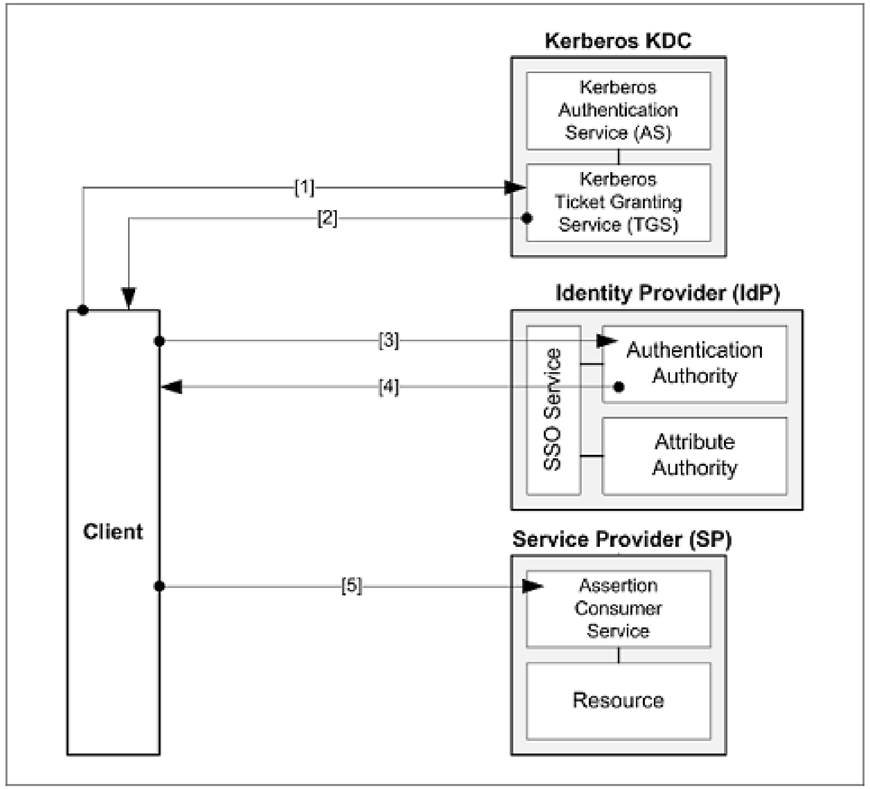
Project Athena

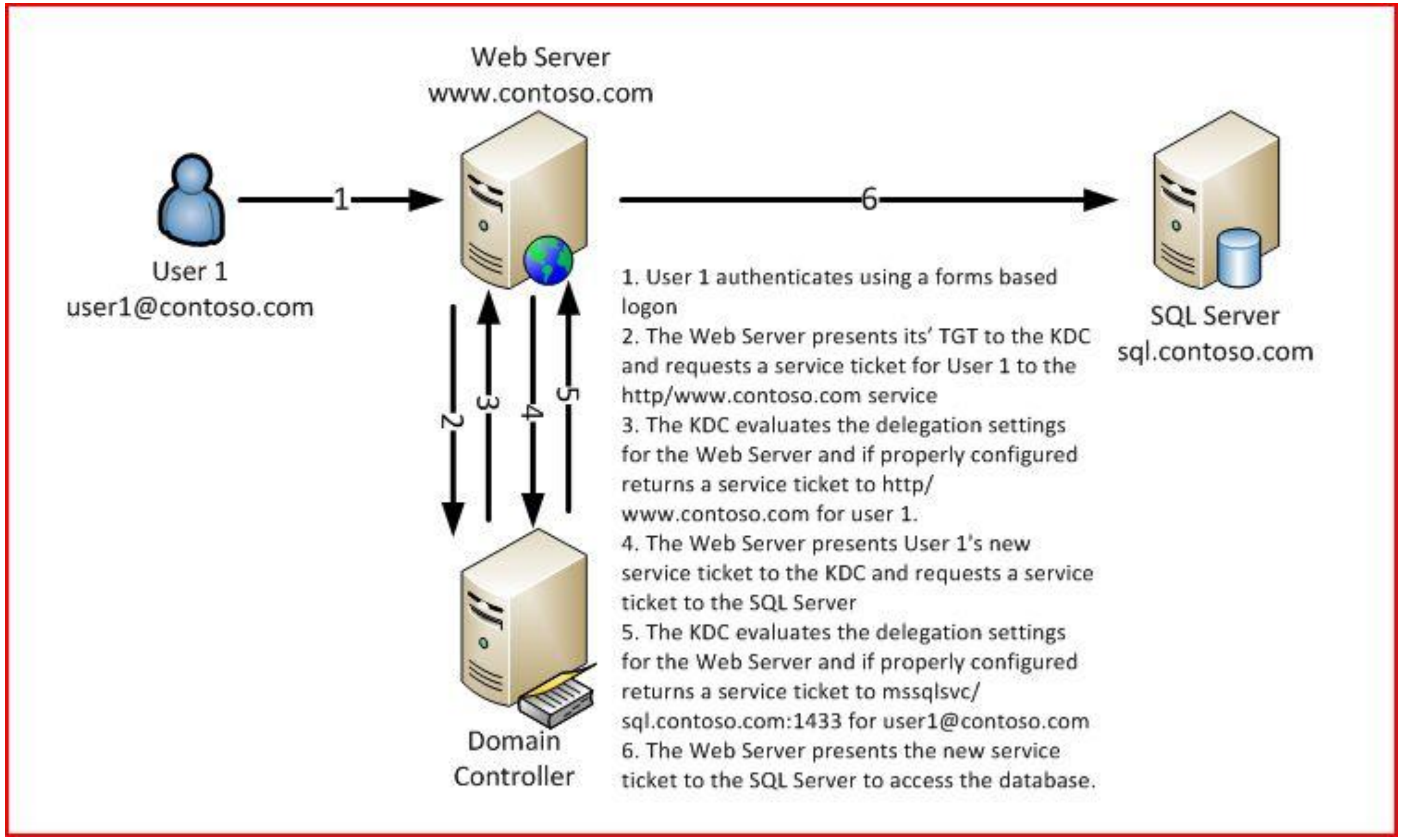
- Started in 1983 by MIT, Digital Equipment Corporation and IBM
- Purpose of producing a campus wide distributed computing environment for educational use.
- Outcomes of Project Athena include : X-Windowing System, Kerberos, Zephyr Notification System

Design Considerations for Kerberos

- No passwords communicated over the n/w.
- Cryptographic protection
- Limited period of validity of sessions
- Timestamping to prevent replay attacks.
- Mutual authentication

Use-case



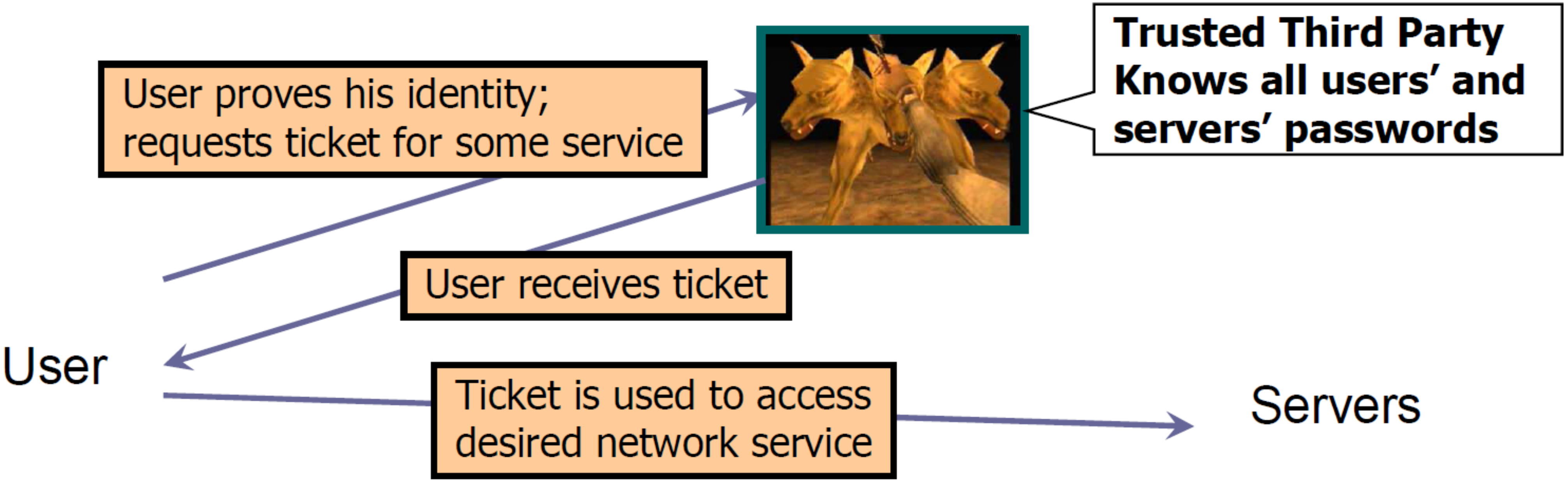


1. User 1 authenticates using a forms based logon
2. The Web Server presents its' TGT to the KDC and requests a service ticket for User 1 to the http/www.contoso.com service
3. The KDC evaluates the delegation settings for the Web Server and if properly configured returns a service ticket to http/ www.contoso.com for user 1.
4. The Web Server presents User 1's new service ticket to the KDC and requests a service ticket to the SQL Server
5. The KDC evaluates the delegation settings for the Web Server and if properly configured returns a service ticket to mssqlsvc/ sql.contoso.com:1433 for user1@contoso.com
6. The Web Server presents the new service ticket to the SQL Server to access the database.

Authenticating to Multiple Servers

- Consider a set of users that needs to access different services on the net
  - Need to authenticate to each of them
  - Naïve solution: every server knows every user's password
    - Insecure: breaking into one server can compromise all users
    - Inefficient: to change password, a user must contact every server

Trusted Third Party



User proves his identity; requests ticket for some service

Trusted Third Party Knows all users' and servers' passwords

User receives ticket

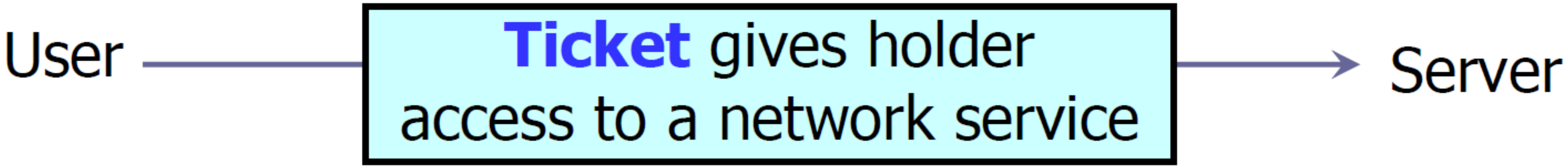
User

Ticket is used to access desired network service

Servers

- Trusted *authentication service* on the network
  - Knows all passwords, can grant access to any server
  - Convenient, but also the single point of failure
  - Requires high level of physical security

What is a ticket?



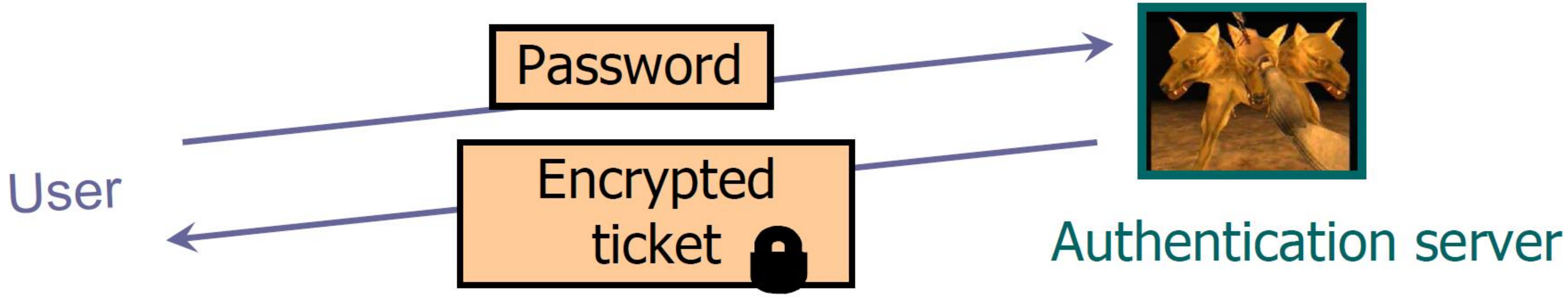
- Ticket cannot include server’s plaintext password
  - Otherwise, next time user will access server directly without proving his identity to authentication service
- Solution: encrypt some information with a key known to the server (but not the user!)
  - Server can decrypt ticket and verify information
  - User does not learn server’s key

Contents of a Ticket

- User name
- Server name
- Address of user's workstation
  - Otherwise, a user on another workstation can steal the ticket and use it to gain access to the server
- Ticket lifetime (duration for which valid)
- A few other things (e.g., session key)

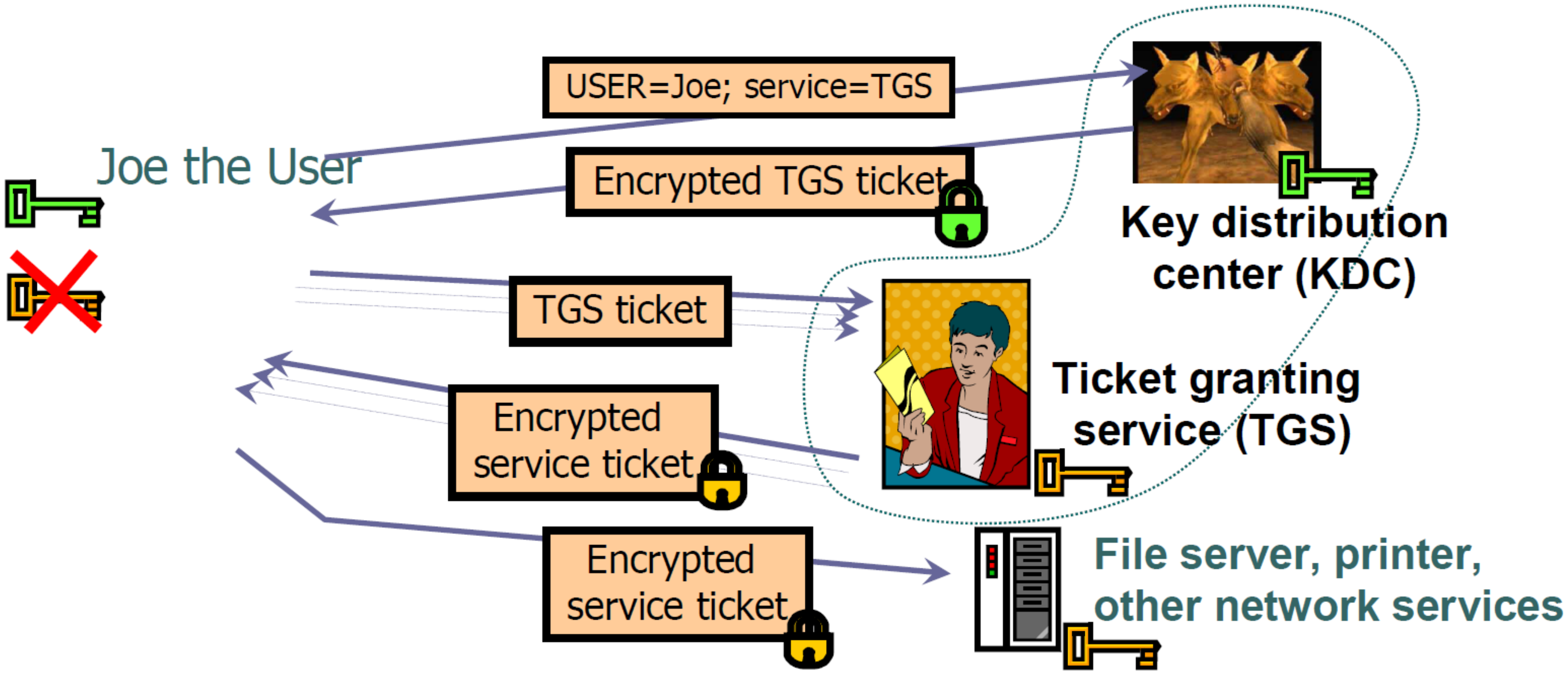
Typically encrypted using the private key of the service or TGS to be accessed.

User Authentication to Third Party



- **Insecure:**
  - **Eavesdropper can steal the password and later impersonate the user to the authentication server**
- **Inconvenient:**
  - **Need to send the password each time to obtain the ticket for any network service**
  - **Separate authentication for email, printing, etc.**

Two-Step Authentication



- Prove identity once to KDC obtain special TGS ticket
  - Use TGS ticket in communications with TGS to get tickets for any network service

Symmetric Keys in Kerberos

- K_c : private key of client C
  - Derived from user's password
  - Known to client and key distribution center (KDC)
- K_{TGS} : private key of TGS
  - Known to KDC and ticket granting service (TGS)
- K_v : private key of network service V
  - Known to V and TGS; separate key for each service
- $K_{c,TGS}$: session key between C and TGS
  - Created by KDC, known to C and TGS, valid only for one session (some lifetime) between C and TGS
- $K_{c,v}$: session key between C and V
  - Created by TGS, known to C and V, valid only for one session (some lifetime) between C and TGS

“Single Logon” Authentication

- Client C types in password once
  - Converted to client key K_c
- C sends to KDC : (IDC, IDTGS, $time_C$)
- KDC sends to C : ($K_{c,TGS}$, ID_{TGS}, $time_{KDC}$, lifetime, $ticket_{TGS}$) encrypted with K_c
  - $ticket_{TGS}$ = ($K_{c,TGS}$, ID_C, $Addr_C$, ID_{TGS} , $time_{KDC}$, lifetime) encrypted with K_{TGS}
  - Client will use this ticket to get other tickets without re-authenticating
- $K_{C,TGS}$: short term session key
  - used for communication between C and TGS during lifetime
- Typical validity of TGS ticket – 1 day
  - Client only needs to obtain TGS ticket once a day (say, every morning)
  - Password is entered once and then deleted from the client machine after obtaining the TGS ticket
  - Password is never sent over the network
  - Ticket is encrypted; client cannot forge it or tamper with it

Obtaining a Service Ticket

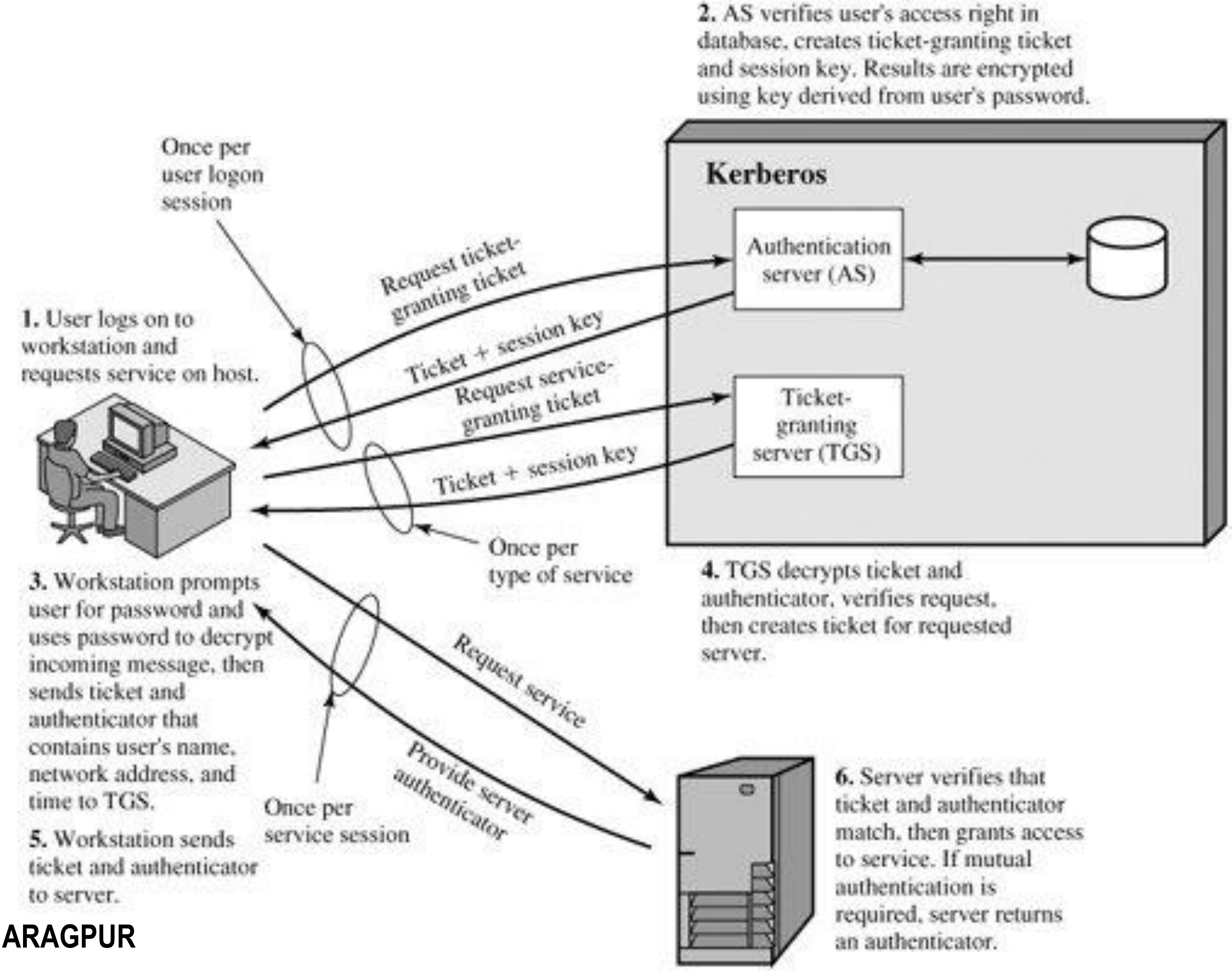
- Client C sends to TGS: (ID_V , $ticket_{TGS}$, $auth_C$)
  - $auth_C$ = (ID_C , $Addr_C$, $time_C$) encrypted with $K_{C,TGS}$
  - authenticator to ensure it is the same client that got the ticket
- TGS sends to C: ($K_{C,V}$, ID_V , $time_{TGS}$, $ticket_V$) encrypted with $K_{C,TGS}$
  - $ticket_V$ = ($K_{C,V}$, ID_C, $Addr_C$, ID_V , $time_{TGS}$, lifetime) encrypted with K_V
- Client uses TGS ticket to obtain a service ticket and a short-term session key for each network service
  - One encrypted, unforgeable ticket per service (printer, email, etc.)

Obtaining Service

- C sends to V: ($ticket_V$, $auth_C$)
- $auth_C$ = (ID_C , $Addr_C$, $time_C$) encrypted with $K_{C,V}$
- V sends to C: ($time_C$+1) encrypted with $K_{C,V}$
  - Authenticates server to client
- For each service request, client uses the short-term session key for that service and the ticket he received from TGS

Summary of Kerberos

1. User is authenticated by the KDC, receives a ticket for the TGS.
2. User uses TGS ticket (typical validity of one day) to request a service ticket from the TGS.
3. User uses Service Ticket to contact service.



2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

1. User logs on to workstation and requests service on host.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos
Authentication server (AS)
Ticket-granting server (TGS)
Once per user logon session
Request ticket-granting ticket
Ticket + session key
Request service-granting ticket
Ticket + session key
Once per type of service
Request service
Provide server authenticator
Once per service session

Important Ideas in Kerberos

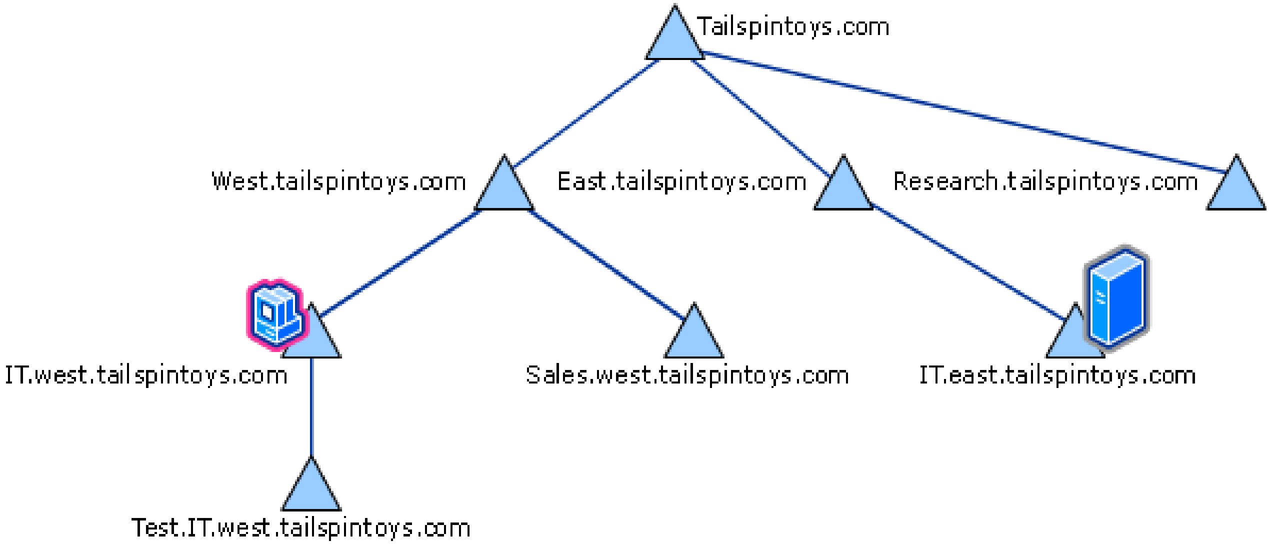
- *Short-term session keys*
  - Long-term secrets used only to derive short-term keys
  - Separate session key for each user-server pair
    - … but multiple user-server sessions re-use the same key
- *Proofs of identity are based on authenticators*
  - Client encrypts his identity, address and current time using a short-term session key
    - Also prevents replays (if clocks are globally synchronized)
  - Server learns this key separately (via encrypted ticket that client can’t decrypt) and verifies user’s identity

Kerberos in Larger Networks

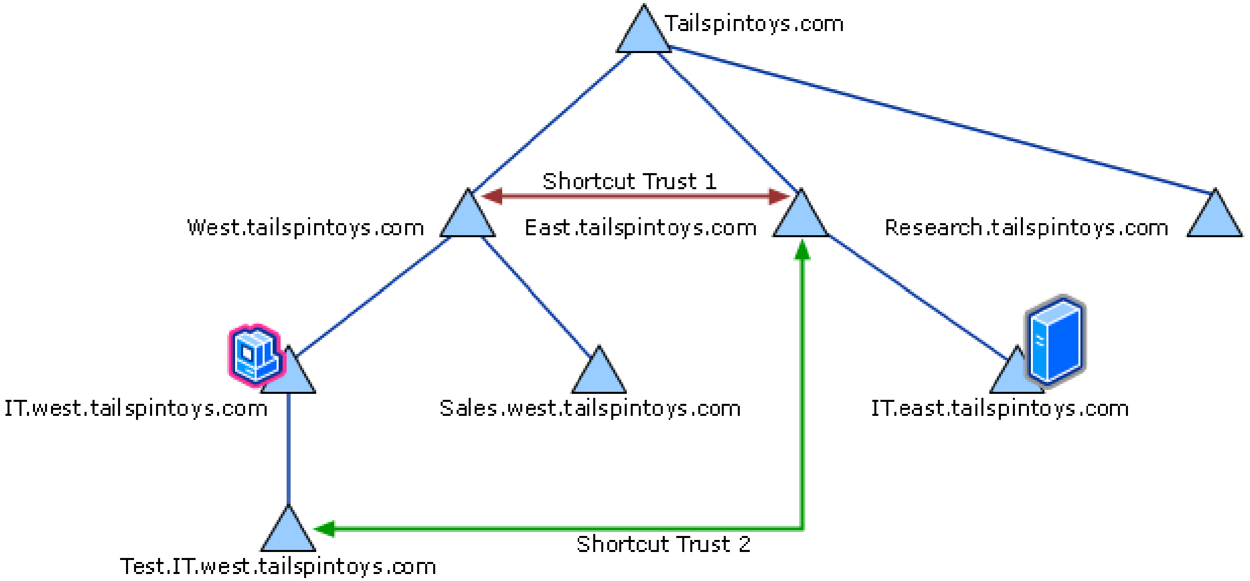
- One KDC isn't enough for large networks (why?)
- Network is divided into realms
  - KDCs in different realms have different key databases
- To access a service in another realm, users must do cross-realm authentication
  - Get ticket for home-realm TGS from home-realm KDC
  - Get ticket for remote-realm TGS from home-realm TGS
  - As if remote-realm TGS were just another network service
  - Get ticket for remote service from that realm's TGS
  - Use remote-realm ticket to access service
  - N(N-1)/2 key exchanges for full N-realm interoperation (NOT SCALABLE)
- Use Hierarchical cross-realm authentication

Hierarchical Cross-realm Authentication

- Organise the realms as trees
- Each node’s TGS knows the TGS key of its parent and children.



Without Shortcut Paths



With Shortcut Paths

Additional Caveats

- Tickets can be of different types
  - Renewabel Ticket:
    - Can be a TGS or Service Ticket
    - Client gets a ticket with renewable flag set and 2 timelimits t_1(Current Expiry) and t_2(Max Expiry)
    - Possibility 1: Ticket expires at t_1 (nothing is done to renew)
    - Possibility 2: Ticket is presented to ticketing agent (KDC or TGS) before t_1 for renewal.

- **Forwardable Tickets**
  - Authentication forwarding is an instance of a proxy where the service that is granted is complete use of the client's identity.
  - The FORWARDABLE flag in a ticket is normally only interpreted by the ticket-granting service. It can be ignored by application servers.
  - With the FORWARDABLE flag TGTs may also be issued with different network addresses.
    - This flag allows for authentication forwarding without requiring the user to enter a password again.
  - The FORWARDED flag is set by the TGS on request.
    - Client supplies a set of addresses for the new ticket. It is also set in all tickets issued based on tickets with the FORWARDED flag set.
  - Application servers may choose to process FORWARDED tickets differently than non-FORWARDED tickets.