Authentication Applications

Authentication Applications

- will consider authentication functions
- developed to support application-level authentication & digital signatures
- will consider Kerberos – a private-key authentication service
- then X.509 directory authentication service

Kerberos

- Network Authentication Protocol
- It provides for strong authentication for client-server applications.
- Uses secret-key cryptography to provide this strong authentication.
- Why Kerberos ???
- Authentication is a key feature in multi-user system
  - divide up resources w/ capabilities between many users
  - restrict user's access to resources.
  - typical authentication mechanism – passwords.

Kerberos

- “ Kerberos is the three-headed dog that guarded the entrance to Hades” –Ancient greek myth.
- Hades => Underworld (where hackers apparently live).

Kerberos

- Part of project Athena (MIT).
- Trusted 3rd party authentication scheme.
- Assumes that hosts are not trustworthy.
- Requires that each client (each request for service) prove it's identity.
- Does not require user to enter password every time a service is requested!

Kerberos Design

- User must identify itself once at the beginning of a workstation session (login session).
- Passwords are never sent across the network in cleartext (or stored in memory)

Kerberos Design (cont.)

- Every user has a password.
- Every service has a password.
- The only entity that knows all the passwords is the *Authentication Server*.

Kerberos

- trusted key server system from MIT
- provides centralised private-key third-party authentication in a distributed network
  - allows users access to services distributed through network
  - without needing to trust all workstations
  - rather all trust a central authentication server
- two versions in use: 4 & 5

Kerberos Requirements

- first published report identified its requirements as:
  - security
  - reliability
  - transparency
  - scalability
- implemented using an authentication protocol based on Needham-Schroeder

Kerberos 4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT

Kerberos Version 4

- Notations:
  - C = Client
  - AS = authentication server
  - V = server
  - ID_c = identifier of user on C
  - ID_v = identifier of V
  - P_c = password of user on C
  - ADc = network address of C
  - K_v = secret encryption key shared by AS an V
  - TS = timestamp
  - || = concatenation

A Simple Authentication Dialogue

(1) C → AS: $IDc \,||\, P_c \,||\, IDv$

(2) AS → C: Ticket

(3) C → V: $IDc \,||\, Ticket$

$Ticket = E_{Kv}[IDc \,||\, P_c \,||\, IDv]$

Version 4 Authentication Dialogue

- Problems:
  - Lifetime associated with the ticket-granting ticket
  - If too short → repeatedly asked for password
  - If too long → greater opportunity to replay
- The threat is that an opponent will steal the ticket and use it before it expires

Version 4 Authentication Dialogue

Authentication Service Exhange: To obtain Ticket-Granting Ticket

(1) C → AS: IDc || IDtgs ||TS1

(2) AS → C: EKc [Kc,tgs|| IDtgs || TS2 || Lifetime2 || Tickettgs]

Ticket-Granting Service Echange: To obtain Service-Granting Ticket

(3) C → TGS: IDv ||Tickettgs ||Authenticatorc

(4) TGS → C: EKc [Kc,¨v|| IDv || TS4 || Ticketv]

Client/Server Authentication Exhange: To Obtain Service

(5) C → V: Ticketv || Authenticatorc

(6) V → C: EKc,v[TS5 +1]

Table 11.1 Summary of Kerberos Version 4 Message Exchanges

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) **C → AS:** $ID_c \| ID_{tgs} \| TS_1$

(2) **AS → C:** $E_{K_c}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2]$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) **C → TGS:** $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) **TGS → C:** $E_{K_{c,tgs}}[K_{c,v} \| ID_v \| TS_4 \| Ticket_v]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2]$

$Ticket_v = E_{K_v}[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4]$

$Authenticator_c = E_{K_{c,tgs}}[ID_c \| AD_c \| TS_3]$

(c) Client/Server Authentication Exchange: to obtain service

(5) **C → K:** $Ticket_v \| Authenticator_c$

(6) **K → C:** $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$Ticket_v = E_{K_v}[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4]$

$Authenticator_c = E_{K_{c,v}}[ID_c \| AD_c \| TS_5]$

Table 11.2 Rationale for the Elements of the Kerberos Version 4 Protocol (page 1 of 2)

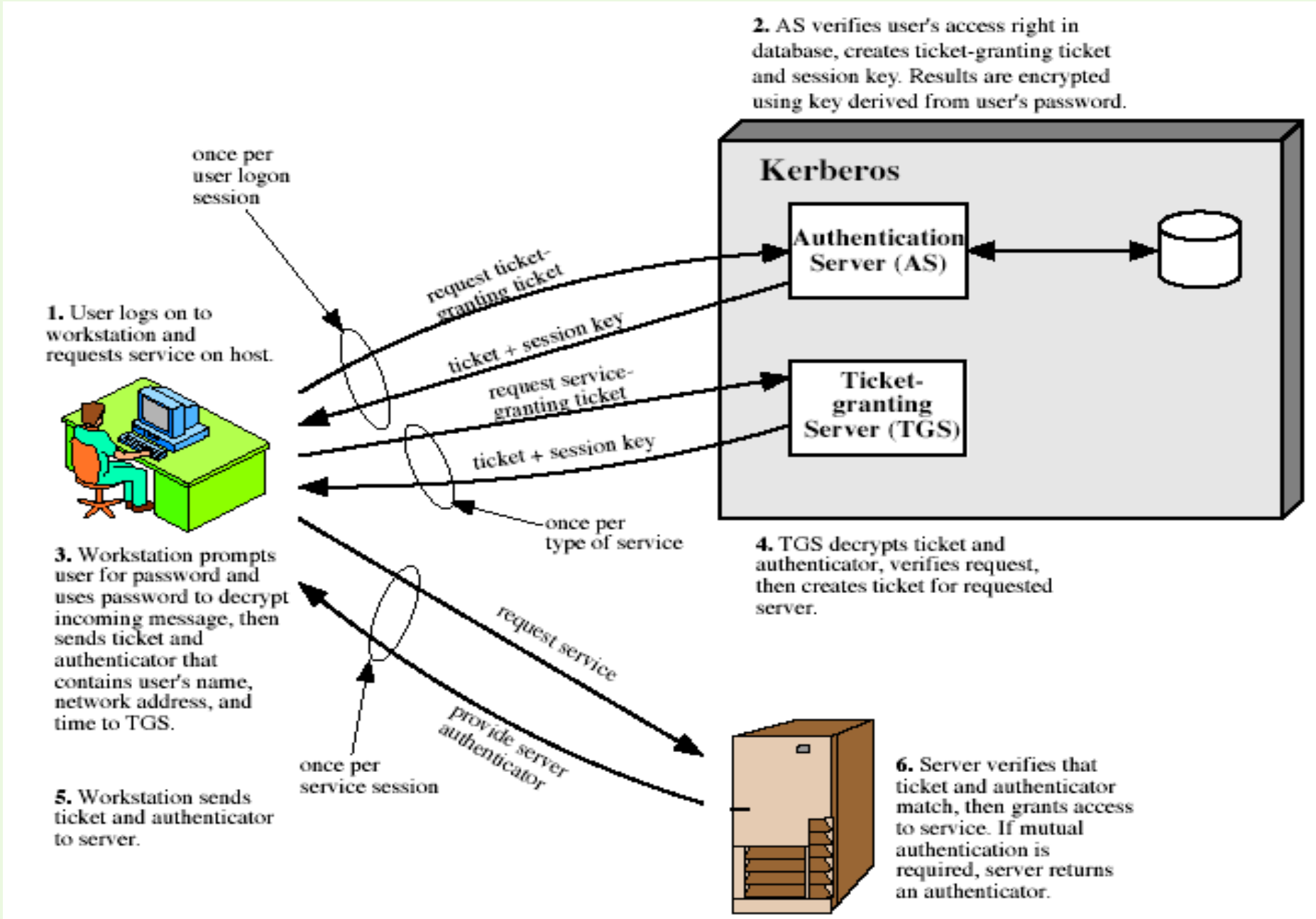
(a) Authentication Service Exchange	
Message (1)	Client requests ticket-granting ticket
ID_C:	Tells AS identity of user from this client
IDtgs:	Tells AS that user requests access to TGS
TS_1:	Allows AS to verify that client's clock is synchronized with that of AS
Message (2)	AS returns ticket-granting ticket
E_{K_C}:	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2)
$K_{c,tgs}$:	Copy of session key accessible to client; created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key
ID_{tgs}:	Confirms that this ticket is for the TGS
TS_2:	Informs client of time this ticket was issued
$Lifetime_2$:	Informs client of the lifetime of this ticket
$Ticket_{tgs}$:	Ticket to be used by client to access TGS
(b) Ticket-Granting Service Exchange	

(b) Ticket-Granting Service Exchange

Message (3)	Client requests service-granting ticket
ID_V:	Tells TGS that user requests access to server V
$Ticket_{tgs}$:	Assures TGS that this user has been authenticated by AS
$Authenticator_c$:	Generated by client to validate ticket
Message (4)	TGS returns service-granting ticket
$E_{Kc,tgs}$:	Key shared only by C and TGS; protects contents of message (4)
$K_{c,tgs}$:	Copy of session key accessible to client; created by TGS to permit secure exchange between client and server without requiring them to share a permanent key
ID_V:	Confirms that this ticket is for server V
TS_4:	Informs client of time this ticket was issued
$Ticket_V$:	Ticket to be used by client to access server V
$Ticket_{tgs}$	Reusable so that user does not have to reenter password
E_{Ktgs}:	Ticket is encrypted with key known only to AS and TGS, to prevent tampering
$K_{c,tgs}$:	Copy of session key accessible to TGS; used to decrypt authenticator, thereby authenticating ticket
ID_c:	Indicates the rightful owner of this ticket
AD_c:	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_{tgs}:	Assures server that it has decrypted ticket properly
TS_2:	Informs TGS of time this ticket was issued
$Lifetime_2$:	Prevents replay after ticket has expired

$Authenticator_c$:	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay
$E_{Kc,tgs}$:	Authenticator is encrypted with key known only to client and TGS, to prevent tampering
ID_c:	Must match ID in ticket to authenticate ticket
AD_c:	Must match address in ticket to authenticate ticket
TS_2:	Informs TGS of time this authenticator was generated

Kerberos 4 Overview



Kerberos Realms

- a Kerberos environment consists of:
  - a Kerberos server
  - a number of clients, all registered with server
  - application servers, sharing keys with server
- this is termed a realm
  - typically a single administrative domain
- if have multiple realms, their Kerberos servers must share keys and trust

Kerberos Version 5

- developed in mid 1990's
- provides improvements over v4
  - addresses environmental shortcomings
    - encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, interrealm auth
  - and technical deficiencies
    - double encryption, non-std mode of use, session keys, password attacks
- specified as Internet standard RFC 1510

Table 11.3 Summary of Kerberos Version 5 Message Exchanges

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) $C \rightarrow AS$: $Options \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) $AS \rightarrow C$: $Realm_c \| ID_c \| Ticket_{tgs} \| E_{Kc}[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[Flags \| K_{c,tgs} \| Realm_c \| ID_c \| AD_c \| Times]$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) $C \rightarrow TGS$: $Options \| ID_v \| Times \| \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C$: $Realm_c \| ID_c \| Ticket_v \| E_{K_{c,tgs}}[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v]$

$Ticket_{tgs} = E_{K_{tgs}}[Flags \| K_{c,tgs} \| Realm_c \| ID_c \| AD_c \| Times]$

$Ticket_v = E_{K_V}[Flags \| K_{c,v} \| Realm_c \| ID_c \| AD_c \| Times]$

$Authenticator_c = E_{K_{c,tgs}}[ID_c \| Realm_c \| TS_1]$

(c) Client/Server Authentication Exchange: to obtain service

(5) $C \rightarrow TGS$: $Options \| Ticket_v \| Authenticator_c$

(6) $TGS \rightarrow C$: $E_{K_{c,v}}[TS_2 \| Subkey \| Seq\#]$

$Ticket_v = E_{K_V}[Flags \| K_{c,v} \| Realm_c \| ID_c \| AD_c \| Times]$

$Authenticator_c = E_{K_{c,V}}[ID_c \| Realm_c \| TS_2 \| Subkey \| Seq\#]$

Table 11.4 Kerberos Version 5 Flags

INITIAL	This ticket was issued using the AS protocol, and not issued based on a ticket-granting ticket.
PRE-AUTHENT	During initial authentication, the client was authenticated by the KDC before a ticket was issued.
HW-AUTHENT	The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client.
RENEWABLE	Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date.
MAY-POSTDATE	Tells TGS that a post-dated ticket may be issued based on this ticket-granting ticket.
POSTDATED	Indicates that this ticket has been postdated; the end-server can check the authtime field to see when the original authentication occurred.
INVALID	This ticket is invalid and must be validated by the KDC before use.
PROXIABLE	Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket.
PROXY	Indicates that this ticket is a proxy.
FORWARDABLE	Tells TGS that a new ticket-granting ticket with a different network address may be issued based on this ticket-granting ticket.
FORWARDED	Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket.

Difference Between Version 4 and 5

- Encryption system dependence (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication