

# Computer Security

Kerberos

# Kerberos

- secret key based
- used for authentication

# Model of User Session

- User logs in
- During session accesses remote resources (servers)

# Servers

- these resources need to authenticate the user
- user's workstation performs authentication protocol on user's behalf
- Do not implicitly trust users
- Have no initial shared secret with users

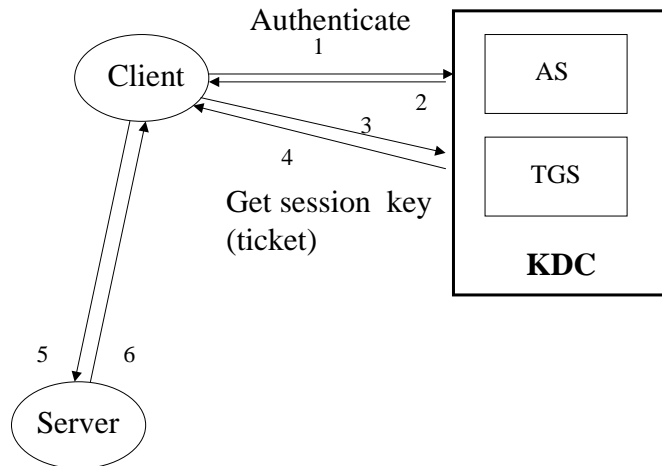
# Kerberos Implementation

- KDC - on physically secure node
- Kerberos consists of a library of subroutines

# KDC

- Authenticates user
- Issues session keys for use between user and servers

# Kerberos



# Use

- user logs on using password and name
- used by workstation to obtain information from KDC

## Keys

- KDC shares a master key with each principal (users and servers)
- KDC invents session keys
- encrypts this with master keys for the two principals

9

## Ticket

- session key encrypted with Bob's secret is a ticket
- Alice can not read ticket
- Alice and Bob can authenticate each other based on session key
- session key plus ticket are Alice's **credentials** to Bob

10

## Password

- workstation could remember password for all of login session
- this leaves password vulnerable
- workstation gets session key from KDC for login session
- then forgets password

11

## Session Key

- KDC generates session key  $S_a$
- transmits it, encrypted with Alice's master key, to her workstation
- KDC also sends **ticket-granting ticket** (TGT) -  $S_a$ , Alice's name, expiration time, etc
- TGT encrypted with KDC's master key

12

## Ticket-Granting Ticket

- used when Alice need to access remote resource
- Alice's workstation transmits TGT to KDC
- also transmits name of the resource

13

## Ticket-Granting Ticket

- TGT contains information KDC needs
- session key, Alice's name, expiration time etc
- KDC does not need any volatile data

14

## KDC

- decrypts TGT to get  $S_a$
- uses  $S_a$  to encrypt session key for Alice
- TGT tells KDC to use  $S_a$  instead of Alice's master key

15

## Authentication Server / Ticket-Granting Server

- in Kerberos documentation KDC consists of
  - Authentication Server
  - Ticket-Granting Server
- both need to know master keys, so often the same thing

16

## Authentication Server/Ticket-Granting Server

- AS gives out Ticket-granting Tickets
- TGS gives out all other tickets

17

## Kerberos

- secret key/DES
- KDC stores master keys encrypted under its own master key

18

## Login

- Alice logs in to workstation
- workstation converts password to DES key
- workstation sends request to KDC
- KDC returns credentials
  - session key  $S_a$
  - ticket granting ticket

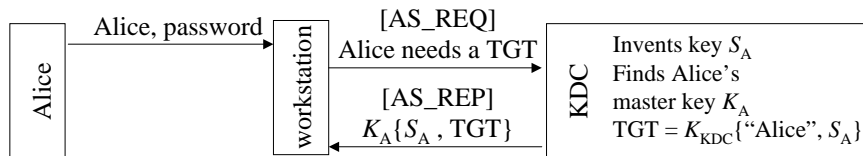
19

## Notes

- ticket doubly encrypted
- this has no security benefit and a slight performance degradation
- V4 does not prompt for password until credentials actually required
- workstation knows password for minimum time possible

20

## Obtaining a TGT



21

## Use of TGT

- Alice may eventually require access to a remote resource/server
- her workstation sends to KDC
  - TGT
  - name of resource/server
  - authenticator

22

## Authenticator

- proves workstation knows session key
- consists of time of day encrypted with session key

23

## Authenticator

- use of authenticator requires nodes to have reasonably synchronised time
- allowable skew set independently at each server
- usually ~5 minutes

24

## Authenticator

- no security or functionality gained by including authenticator when asking for ticket
- done to make protocol for talking to TGS same as for talking to other resources

25

## Authenticator

- when talking to other resources authenticator does provide security
- prevents replay
- authenticates sender

26

## KDC

- gets  $S_a$  from TGT
- checks expiration time in TGT
- constructs  $K_{ab}$  and ticket
- encrypts ticket with master key for resource/server

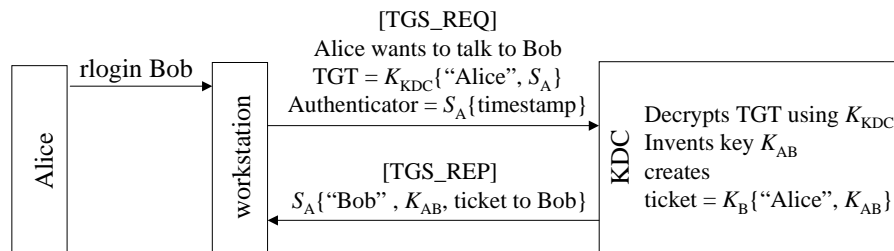
27

## KDC Reply

- ticket for resource/server (eg., Bob)
- session key ( $K_{ab}$ )
- all encrypted with  $S_a$

28

## Obtaining Ticket



29

## Request to Bob

- ticket
- authenticator (time encrypted with  $K_{ab}$ )

30

## Server Action

- Bob decrypts ticket - get  $K_{ab}$
- use  $K_{ab}$  to decrypt authenticator
- checks enclosed time is acceptable

31

## Quick Replays

- could keep all recently received timestamps
- V4 does not
- keeping them does not help with replicated servers anyway

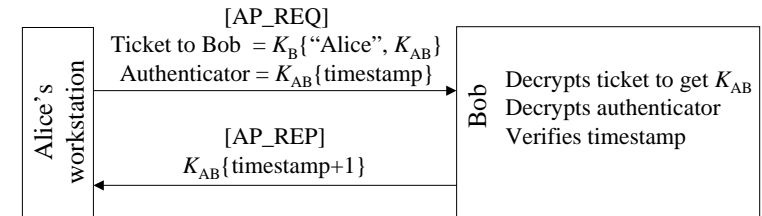
32

## Reply from Bob

- Bob adds one to time
- reencrypts with  $K_{ab}$
- sends it back
- this gives mutual authentication

33

## Server Reply



34

## Replicated KDCs

- single KDC point of failure & performance bottleneck
- can have multiple replicated KDCs
- all share same KDC master key

35

## Consistency

- kept consistent by having one site hold master copy
- updates (addition/modification/deletion) done there
- others periodically download

36

## Transfer

- could be altered/read in transit
- remember keys kept encrypted under KDC master key
- so data sent as is
- followed by cryptographic hash for integrity, with timestamp

37

## Master Copy

- still single point of failure
- but most operations read only
- so not as bad

38

## The Wider World

- one KDC master for everyone is not realistic
- too large
- every copy would reveal all keys

39

## Realms

- the principals in the network are divided into realms
- each realm has its own KDC database
- may be replicated within a realm

40

## Principal Names in V4

- name
- instance (often identifies machine for service or null or role for humans)
- realm

## Inter-Realm Authentication

- principals in one realm may need to authenticate principals in another realm
- KDC in realm B can be registered as a principal in realm A

## Inter-Realm Authentication

- Alice, realm Wonderland
- Dorothy, realm Oz
- Alice's workstation notes Dorothy is in a different realm
- asks KDC for a ticket to Oz's KDC

## Wonderland's KDC

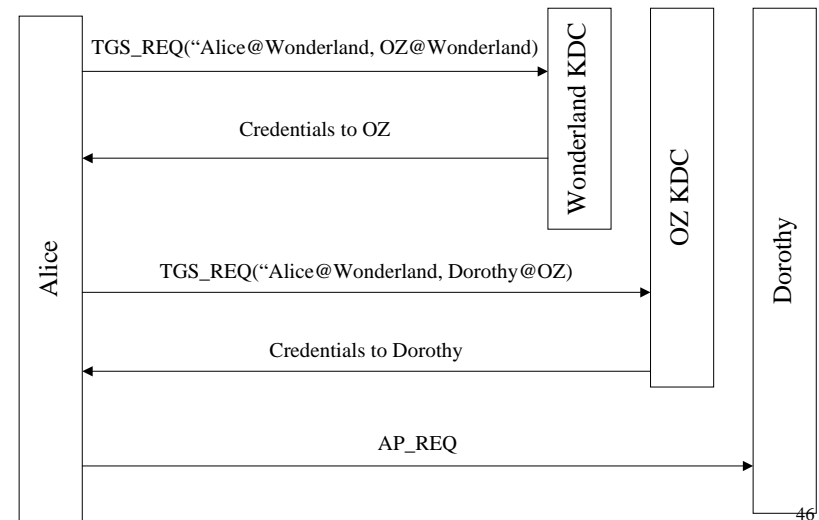
- if Wonderland and Oz have decided to allow inter-realm authentication
- Wonderland KDC gives Alice ticket to Oz's KDC
- encrypted under key agreed between KDC's

## Request to Oz

- Alice sends ticket to Oz's KDC
- includes her realm
- lets Oz's KDC know which key to use to decrypt
- OZ KDC issues Alice a ticket to talk to Dorothy
- includes key  $K_{ad}$

45

## Inter-Realm Authentication



46

## Chaining

- A and B KDC's allow cross-authentication
- B and C KDC's allow cross-authentication
- can user in A talk to user in C?
- not in V4
- stops rogue KDC impersonating everybody

47

## Changing Passwords

- users can change passwords
- unfortunately this also changes user's master secret
- as tickets encrypted with these keys this is a problem

48

## Key Version Numbers

- requiring new tickets could prove inconvenient
- instead each key has a version number
- principals required to remember several versions of key
- tickets expire in ~21 hours, so don't have to remember for long

49

## Humans

- not required to remember different passwords
- login allowed to fail if new password not yet propagated
- they can wait and try again

50

## Kerberos V5

- major overhaul of V4
- same basic philosophy
- major changes to encodings
- major extensions to functionality

51

## Names

- name
- realm

52

## Name

- type
- varying number of arbitrary strings
- so serves purpose of name and instance in V4

## Realm

- DNS standard name or
- X.500 name
- or other

## Delegation

- giving some else access to things you are authorised to access
- usually limited in time and/or scope

## Delegation

- Alice could send Bob her master key
- not desirable

## Delegation

- Alice could obtain tickets
- send these (or TGT) to Bob
- inconvenient, possibly insecure and will not work in Kerberos
- Bob does not have Alice's network layer address

57

## Delegation

- in V5 Alice can ask for a TGT with a network layer address different from hers
- can have multiple or no address in TGT

58

## Outline

- Alice logs in as in V4
- gets session key and TGT with her own network layer address
- later decides she wants Bob to act on her behalf
- asks for TGT with Bob's network layer address
- this TGT not usable by Alice herself

59

## Alternatives

- Kerberos could have put network layer address in authenticator rather than ticket
- then would not need to go back to KDC
- so alternative chosen has performance disadvantage
- has advantage in that KDC can audit delegation events

60

## Limited Delegation

- Alice may know exactly which rights she wishes to delegate to Bob
- V5 allows two forms of limited delegation
  - giving specific tickets rather than a TGT
  - Alice can request the addition of an AUTHORIZATION-DATA field to a ticket or TGT

61

## AUTHORIZATION-DATA

- not interpreted by Kerberos
- application specific
- tells application what Bob is allowed to do with ticket
- if field is in TGT it is copied into all tickets obtained using that TGT

62

## Delegation

- not universally regarded as a good idea
- so optional in V5
- flag in TGT indicates whether request for TGT or ticket with different network layer address should be allowed

63

## Flags

- Kerberos does not specify how KDC should set flags in initial TGT
- can put instructions in user's entry in KDC database

64

## Flags in Delegation

- is TGT forwardable?
- is TGT proxiable?

## Forwarding

- means TGT can be exchanged for one with a different (Bob's) network layer address
- Bob can then request tickets on Alice's behalf

## Forwarding

- when forwardable TGT used to request a new TGT Alice specifies how forwardable flag set in new TGT
- if set Bob can obtain TGT for (eg.) Carol to use on Alice's behalf

## Proxiable

- TGT can be used to obtain tickets for use with network layer address different to that in the TGT
- can not get a TGT this way
- such tickets referred to as proxy tickets

## Flags

- TGTs have a forwarded flag
- tickets have forwarded and proxy flags
- forwarded flag will be set in any ticket obtained using a forwarded TGT

## Flexibility

- some applications may refuse forwarded and/or proxied tickets
- allowing both KDC and applications to make access control decisions is flexible but confusing

## Ticket Lifetimes - V4

- maximum is about 21 hours
- four octet start time
- one octet lifetime
- units of five minutes
- this is too short for some applications

## Ticket Lifetimes - V5

- effectively unlimited
- farthest in the future that can be specified is 31st December, 9999
- 17 bytes long
- units in seconds
- can also add a microsecond time

## Long-Lived Tickets

- can be security risks
- how to revoke?
- can invalidate master key of service
- but may not want to do that

73

## Timestamp fields

- each stored in 17 octets
- START-TIME
- END-TIME
- AUTHTIME (time TGT issued)
- RENEW-TILL (latest legal end-time)

74

## Renewable Ticket

- may not want to give out a ticket valid for, eg., 100 years
- can give a ticket valid for 100 years if it is renewed, eg., daily
- ticket is given back to KDC
- KDC reissues it with new END-TIME
- unless there is some reason to revoke user's privileges

75

## KDC

- configured with maximum validity for a ticket (eg., a day)
- if ticket is to be valid for longer renewable flag is set
- renew-time sets limit on renewing

76

## User

- must renew ticket before it expires
- this is inconvenient
- must check tickets (manually or by daemon)

77

## Renewing Expired Tickets

- would have been more convenient for user
- could be renewed when presented
- Kerberos does not do this as KDC would have to remember expired ticket until RENEW-TILL time

78

## Postdated Tickets

- START-TIME can be any time in the future
- if start time is in the future, invalid flag set
- ticket represented to KDC for actual use
- allows tickets to be revoked before use

79

## Key Versions

- KDC in V5 also needs to remember past versions of keys
- needs to do this for renewable tickets
- unfortunately, post-dated tickets complicate the amount of time old versions have to be remembered

80

## Different Keys in Different Realms

- humans like to use one password
- if keys the same in each realm then intruder getting key in one realm can impersonate use in other realms
- in V5 password to key hash uses realm name
- this does not help if intruder learns user's password

81

## Multiple Realms -V4

- user in realm A wishes to be authenticated in realm B
- in V4 realm B's KDC must be registered as a principal with realm A's KDC
- with n realms cross-authenticated the management becomes problematic

82

## Multiple Realms - V5

- in V5 it is allowable to go through a series of realms (unlike V4)
- A to B then to C
- but allows KDC in B power to impersonate anyone
- tickets include a TRANSITED field
- lists names of all realms transited to obtain ticket

83

## Transitted Field

- can still do some faking
- but last KDC must be named in transited field
- as next KDC will reject if name does not match key used to encrypt

84

## Trust

- realms might be untrustworthy for transit
- but trustworthy for their own principals

85

## Hierarchy of Realms

- typical policy is to arrange realms in a hierarchy
- realms share keys with their children and parent
- trusted path is shortest route in tree

86

## Cross Links

- sometime desirable to shortcut hierarchy
- efficiency
- trust (bypassing untrusted realm)
- such links called **cross links**
- cross links used if they make path shorter

87

## Password Guessing

- in V4 any user can ask for a TGT for any other user
- no way for KDC to know who is asking
- allows password guessing
- in V5 request for TGT must be accompanied by current timestamp encrypted using user's master key

88

## Password Guessing

- user can use own TGT to get ticket for another principal
- can then try password guessing on ticket
- KDC will not issue tickets to human users

89

## Detailed Protocol

- Kerberos V5
- Refer to slides 21, 29, 34

90

## Authenticator

- Appears in TGS\_REQ and AP\_REQ
- Always encrypted by key in accompanying ticket

AUTHENTICATOR-VNO	Version number (5)
CNAME, CREALM	Alice's name and realm
CKSUM	(optional) checksum of application data that might have been sent along with the AP_REQ
CTIME, CUSEC	Time at Alice (in seconds, microseconds)
SUBKEY	(optional) key Alice would like to use, instead of the key in the ticket, for the conversation with Bob
SEQ-NUMBER	Initial sequence number that Alice will use in her KTB_SAFE and KRB_PROT messages to Bob
AUTHORIZATION-DATA	Application-specific data limiting Alice's rights

91

## Ticket

- Used in TGS\_REQ, AS\_REP, TGS\_REP, KRB\_CRED

MSG-TYPE	Message type (1)
TKT-VNO	Version number (5)
REALM, SNAME	Bob's name and realm
The remainder of the fields are encrypted with Bob's master key	
FLAGS	FORWARDABLE, PROXIABLE, PROXY, MAY-POST-DATE, POSTDATED, INVALID, RENEWABLE, INITIAL (ticket was issued using AS_REQ rather than TGS_REQ) PRE-AUTHENT (user authenticated to the KDC before ticket was issued) HW-AUTHENT (user was authenticated before ticket was issued, using something like a smart card)
CNAME, CREALM	Alice's name and realm
TRANSITED	Names of realms transited between Alice's realm and Bob's realm
AUTH-TIME, START-TIME, END-TIME, RENEW-TILL	Timestamps, START-TIME and RENEW-TILL are optional.
CADDR	(optional) the set of addresses from which this ticket will be valid
AUTHORIZATION-DATA	Application-specific data limiting Alice's rights

92

## AS\_REQ

MSG-TYPE	Message type (10)
PVNO	Protocol version number (5)
PADATA	(optional) preauthentication data – timestamp encrypted with Alice's master key
KDC-OPTIONS	Flags – each flag indicates a request to set the corresponding flag in the ticket the KDC will return (see text, p. 358)
CNAME	Alice's name (c stands for client)
SNAME	Bob's name (s stands for server) will be krbtgt if the request is for a TGT
REALM	Realm in which both Alice and Bob reside
FROM	(postdated ticket) desired start time
TILL	Desired end time, which is the expiration time in the ticket
RTIME	Desired renew-till time (only in request for renewable ticket)
NONCE	Number to be returned in reply to prevent replay attacks (MIT implementation uses current timestamp as the nonce)
ETYPE	Type of encryption Alice would like KDC to use which encrypting the credentials
ADDRESSES	Network layer address to include in ticket – used in proxy or forwardable tickets, or when Alice has multiple network layer addresses

93

## TGS\_REQ

MSG-TYPE	Message type (12)
PVNO	Protocol version number (5)
PADATA	Ticket and authenticator
KDC-OPTIONS	Flags from AS_REQ, plus others (see text p. 359)
SNAME	Bob's name (s stands for server) - will be krbtgt if request is for a TGT
REALM	Realm in which Bob resides
FROM	(postdated ticket) desired start time
TILL	Desired end time, which is the expiration time in the ticket
RTIME	Desired renew-till time (only in request for renewable ticket)
NONCE	Number to be returned in reply to prevent replay attacks (MIT implementation uses current timestamp as the nonce)
ETYPE	Type of encryption Alice would like KDC to use which encrypting the credentials
ADDRESSES	Network layer address to include in ticket – used in proxy or forwardable tickets, or when Alice has multiple network layer addresses
AUTHORIZATION-DATA	Application-specific data to be copied into TGT and tickets requested using that TGT, intended to convey restrictions on use. This field encrypted and integrity protected
ADDITIONAL-TICKETS	Bob's TGT in the case where Bob does not know his master key

94

## AS\_REP

MSG-TYPE	Message type (11)
PVNO	Protocol version number (5)
PADATA	(optional) salt to combine with user's password in order to compute the master key derived from user's password
CREALM	Alice's Realm
CNAME	Alice's name – name and realm let Alice's workstation know what key to use to decrypt
TICKET	The ticket to Bob that Alice requested
ENC-PART	Encrypted portion (see next slide)

95

## AS\_REP encrypted part

KEY	Encryption key associated with the ticket enclosed in the AS_REP
LAST_REQ	A sequence of from 0 to 5 timestamps specifying such information as when Alice last requested a TGT, or last requested any ticket. Not always used – see text p. 361
NONCE	The nonce copied from AS_REQ
KEY-EXPIRATION	(optional) time when user's master key will expire for the purpose of warning Alice to change her password
FLAGS	A copy of the flags that appear inside the ticket (so that Alice can check if the KDC granted all her request, and also allows her to detect malicious modification to AS_REQ)
AUTH-TIME, START-TIME, END-TIME, RENEW-TILL	Timestamps; START-TIME and RENEW-TILL are optional
SREALM, SNAME	Bob's name and realm
CADDR	(optional) the set of addresses from which this ticket will be valid

96

## TGS\_REP

- Very similar to AS\_REP
  - Never a PADATA field in a TGS\_REP
  - Never a KEY-EXPIRATION field in a TGS\_REP
  - ENC-PART field is encrypted with the key in the TGT or ticket sent in the TGS\_REQ (or sub-key in authenticator)

## AP\_REQ

MSG-TYPE	Message type (14)
PVNO	Protocol version number (5)
AP-OPTIONS	Flags, of which two are defined USE-SESSION-KEY, which means the ticket is encrypted under the session key in Bob's TGT, rather than Bob's master key MUTUAL-REQUESTED, which tells Bob mutual authentication is requested
TICKET	The ticket to Bob
AUTHENTICATOR	An authenticator, proving that Alice knows the key inside the ticket

## AP\_REP

MSG-TYPE	Message type (15)
PVNO	Protocol version number (5)
The rest is encrypted	
CTIME	The time copied from the CTIME field of the authenticator in the AP_REQ
CUSEC	The low order bits of CTIME, since CTIME is expressed in seconds this field specifies microseconds
SUBKEY	An optional field intended for Bob to be able to influence the Alice-Bob session key in an application specific way
SEQ-NUMBER	Starting sequence number for messages sent from Bob to Alice