KERBEROS

System Design in V4

Kerberos V4 and V5

- Designed at MIT based on work by Needham and Schroeder
- Private key system using KDCs
- V4 larger installed base, V5 greater functionality
- V4 works only on TCP/IP networks

Key Distribution Centre (KDC)

- Runs on physically secure node
- Library of subroutines
- Database largely static
- Allows authorised users to access securely network resources
- Underlying network assumed insecure
- KDC subroutines called by TELNET (RFC 854), NFS (RFC 1094) and other applications

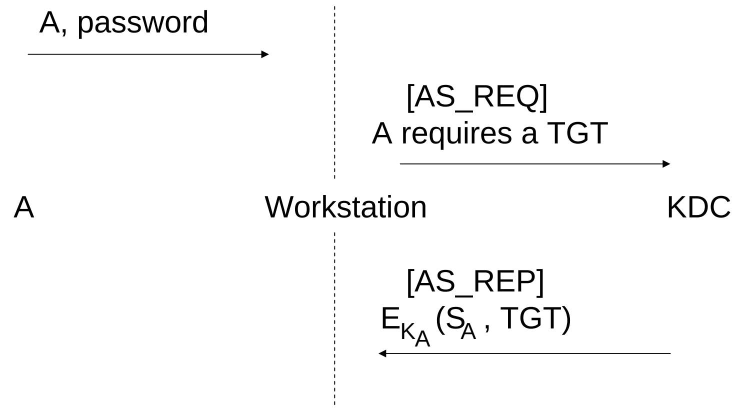
Terminology

- Network users and resources are *Principals* to the KDC
- KDC has *Master Key* for each *Principal*
- *Master Key* is derived from *Password*
- *Master Key* is used to distribute *Session Keys*

Obtaining a TGT (I)

- At login (username, password) A requests session key S_A from KDC
- S_A has limited lifetime (a few hours)
- KDC sends $\{S_A, TGT\}$ encrypted under K_A
- TGT is $\{S_A$, A, expiration time$\}$ encrypted under K_{KDC}
- KDC "forgets" TGT, S_A etc
- On receipt of S_A A "forgets" password

Obtaining a TGT (II)

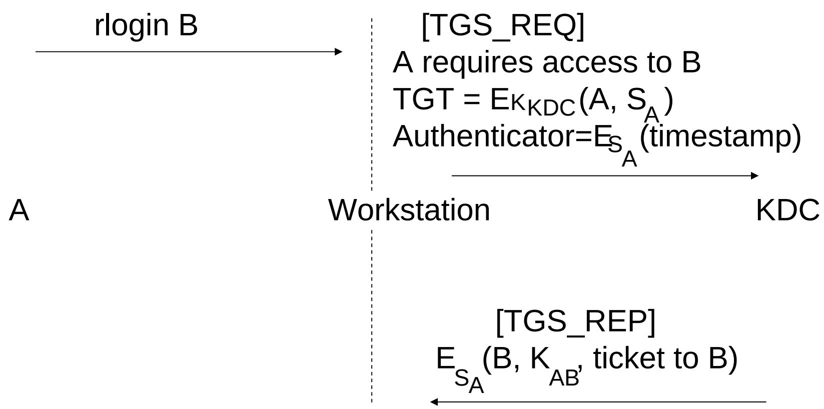


KDC generates key S_A, locates A's master key K_A and creates TGT = $E_{K_{KDC}}(A, S_A)$

Obtaining a ticket for remote login (I)

- A needs access to B
- A sends B, TGT to KDC with an authenticator
- Authenticator is timestamp encrypted under S_A
- KDC sends B, K_{AB} and ticket for B encrypted under S_A
- Authentication Server, Ticket Granting Server and KDC are same resource

Obtaining a ticket for remote login (II)



KDC generates K_{AB}, decrypts TGT to obtain S_A, decrypts authenticator, verifies timestamp, finds K_B, creates ticket = $E_{K_B}(A, K_{AB})$

Remote Login (I)

- A sends ticket to B with authenticator
- Authenticator is timestamp encrypted under K_{AB}
- B decrypts ticket to obtain K_{AB} , decrypts authenticator and verifies timestamp
- B replies to A with an authenticator
- Authenticator is timestamp + 1 encrypted under K_{AB}

Remote Login (II)

[AP_REQ]
ticket to B = $E_{K_B}(A, K_{AB})$
authenticator = $E_{K_{AB}}(\text{timestamp})$

A's Workstation → B

[AP_REP]
$E_{K_{AB}}(\text{timestamp + 1})$

B → A's Workstation

Timestamps and authenticators

- Timestamps protect against replay
- Time skew maximum is 5 minutes
- Mutual authentication by adding 1 to timestamp
- Authenticator in request for ticket adds no security

KDC Configuration

- Database (principal, master key) encrypted under KDC master key
- Kerberos V4 uses DES (V5 supports other algorithms)

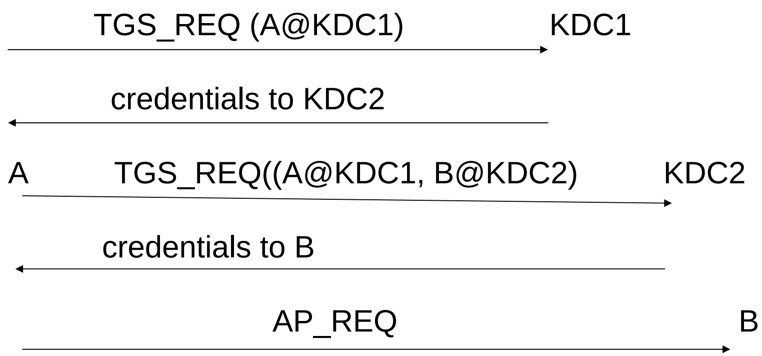
Replicated KDCs

- KDC single point of failure and potential performance bottleneck – replicate KDCs
- One KDC holds master copy
- Master KDC failure impacts only add/deletes and password changes
- Updating slave KDCs presents security issues
- Disclosure protected by KDC master key encryption and integrity by cryptographic hash of file

Realms and Names

- Universal KDC would require universal trust
- Each realm has own KDC database
- Principals have [Name, Instance, Realm]
- Instance is machine running named application
- For human users Instance could indicate role

Inter Realm Communications



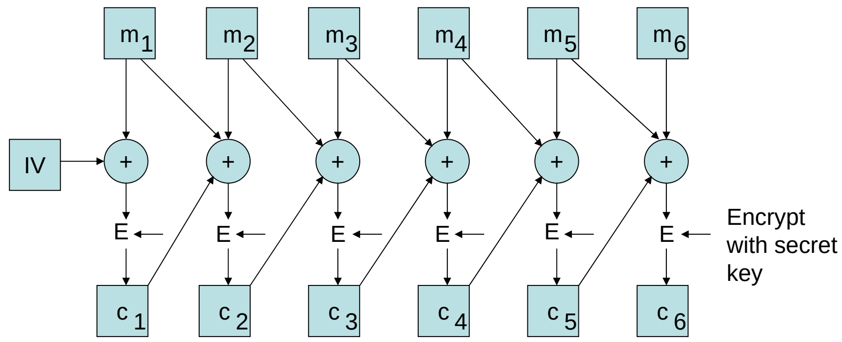
Key Version Numbers

- Master keys change with password
- Keys given version number
- All network resources must remember several versions
- Human users may need to use old password with slave KDCs immediately after logging change with master KDC

Encryption for privacy and integrity

- Standard method (not in V4) is CBC for encryption and CBC residue (with different key) for integrity
- Integrity alternative is to add redundant plaintext before encryption and check for match after decryption – most such schemes are flawed
- V4 uses plaintext CBC (PCBC) – not totally secure

Plaintext Cipher Block Chaining (PCBC)



Encryption for Integrity

- V4 uses variation on checksum algorithm devised by Jueneman
- Checksum formed by hashing message S_A
- Details not published
- Not adopted in V5

Network Layer Addresses in Tickets

- When A requests a ticket for B, KDC adds A's network address to ticket
- B compares address in ticket to connect request
- Protects against impersonation
- Prevents delegation