Kerberos: An Authentication Service for Computer Networks

by Clifford Neuman and Theodore Ts'o

Presented by:

Smitha Sundareswaran

Chi Tsong Su

Introduction

- Kerberos: An authentication protocol based on cryptography
- Designed at MIT under project Athena
- Variation of Needham Schroeder protocol

  - *Difference: Kerberos assumes all systems on the network to be synchronized*
- Similar function as its mythological namesake: “guards” the access to network protocols

Contribution

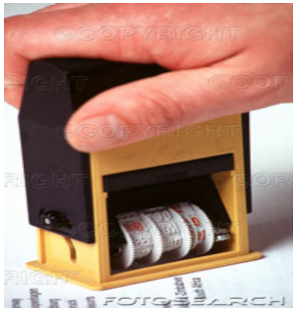
- Defines ideas of authentication, Integrity, confidentiality and Authorization
- Working of Kerberos
- Limitations
- Utilities
- How to obtain and use Kerberos
- Other methods to improve security

Why Kerberos?

- Foils threats due to eavesdropping
- More convenient than password based authentication
  - Allows user to avoid "*authentication by assertion*"
- Authentication based on cryptography: attacker can't impersonate a valid user

How Kerberos Works

- Distributed authentication service using a series of encrypted messages
    - Password doesn’t pass through the network
- Timestamps to reduce the number of messages needed for authentication
- “Ticket granting Service” for subsequent authentication



Kerberos Authentication and Encryption

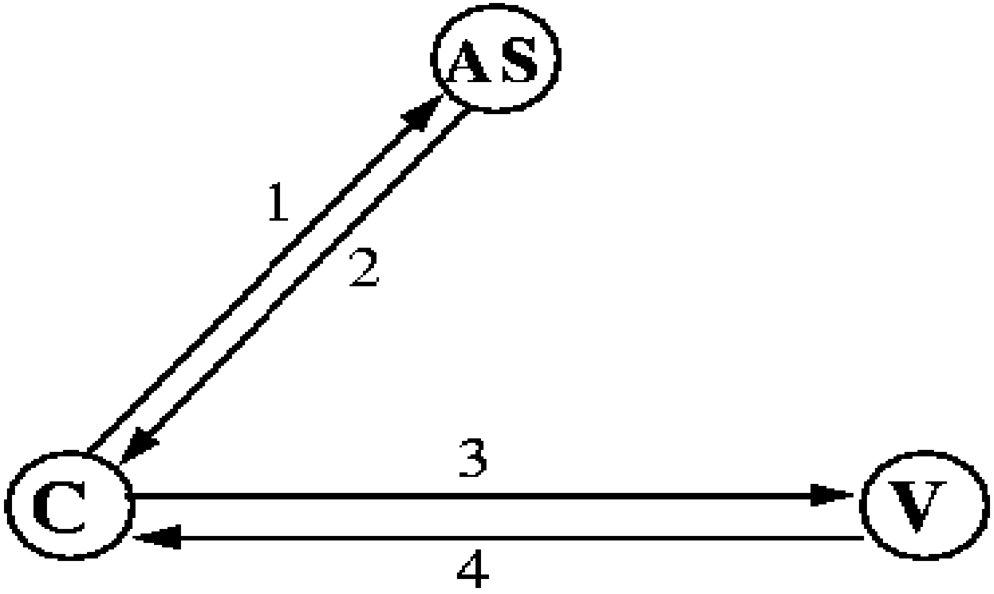
- Authentication proves that a client is running on behalf of a particular user
- Uses encryption key for authentication
  - Encryption key = Password
- Encryption implemented using DES
  - Checksum included in message checksum and encryption provide integrity & confidentiality

The Kerberos Ticket

- Initially, client and Server don’t share an encryption key
- Authentication server generates an encryption key (session key) and distributes it to client and verifier
- Kerberos Ticket is a certificate issued by authentication server used to distribute the session key
- Ticket = session key + name of principal + expiration time for key

Basic Kerberos Protocol

Application request and response

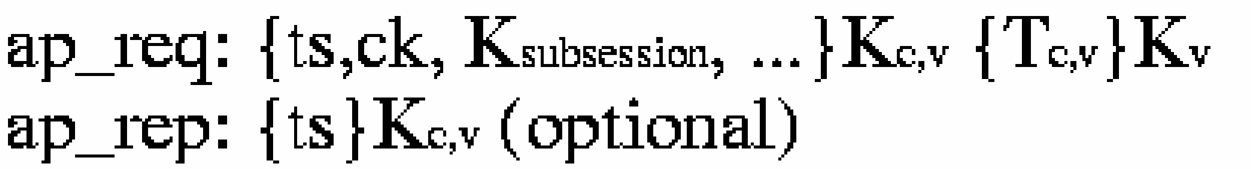


1. as_req: c, v, $time_{exp}$, n
2. as_rep: $\{K_{c,v}, v, time_{exp}, n, \ldots\}K_c$, $\{T_{c,v}\}K_v$
3. ap_req: $\{ts, ck, K_{subsession}, \ldots\}K_{c,v}$ $\{T_{c,v}\}K_v$
4. ap_rep: $\{ts\}K_{c,v}$ (optional)

$T_{c,v} = K_{c,v}, c, time_{exp} \ldots$

Application request and response (cont’d.)

- Most Basic exchange of the protocol



ap_req: $\{ts, ck, K_{subsession}, \ldots\}K_{c,v}\ \{T_{c,v}\}K_v$

ap_rep: $\{ts\}K_{c,v}$ (optional)

- Used by client to prove to verifier that it knows the session key embedded in a ticket
- Application request = ticket + authenticator
- Authenticator : {current time, checksum, optional encryption key,..} encrypted with session key

Authentication request and response

- Used when client requires association with particular verifier
- Request : $\mathrm{as\_req}: c, v, \mathrm{time}_{exp}, n$
- Response: $\mathrm{as\ \ rep}: \{K_{c,v}, v, \mathrm{time}_{exp}, n, \ldots\}K_{c}, \{T_{c,v}\}K_{v}$

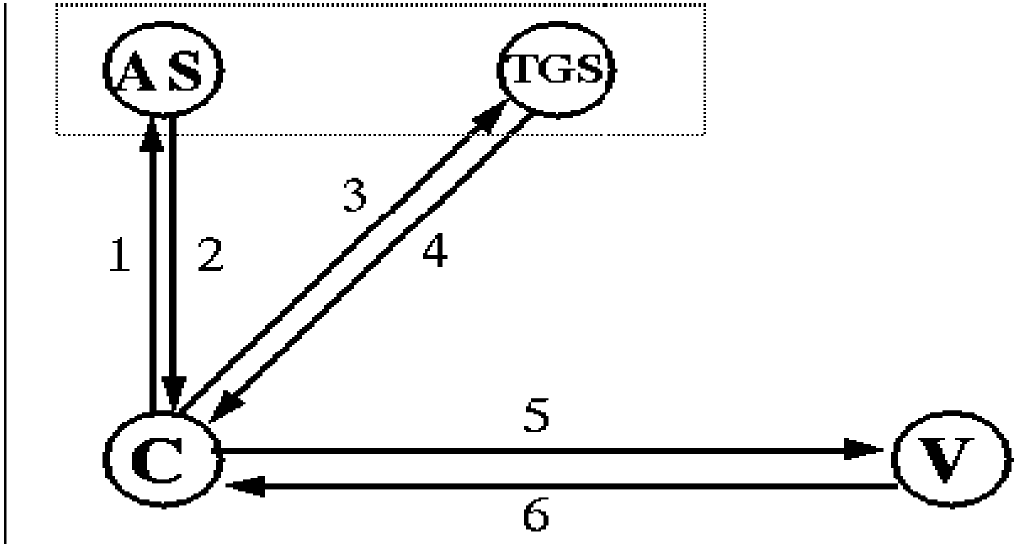
Obtaining additional Tickets

- Basic Kerberos protocol requires user’s password to be presented every time for authentication to new verifier

  - *Cumbersome!!*
- Ticket granting Exchange used to support single sign-on using short lived credentials

  - Credentials (tickets and encryption keys are cached)
- Authentication request gets a *ticket granting ticket* and session key in response from authentication server
- For subsequent authentication, a new ticket is request from authentication server using the ticket granting exchange

Complete Kerberos Authentication Protocol



1. as_req: c, tgs, time_{exp}, n
2. as_rep: $\{K_{c,tgs}, tgs, \text{time}_{\text{exp}}, n, \ldots\}K_{c}, \{T_{c,tgs}\}K_{tgs}$
3. tgs_req: $\{ts, \ldots\}K_{c,tgs}\ \{T_{c,tgs}\}K_{tgs}, v, \text{time}_{\text{exp}}, n$
4. tgs_rep: $\{K_{c,v}, v, \text{time}_{\text{exp}}, n, \ldots\}K_{c,tgs}, \{T_{c,v}\}K_{v}$
5. ap_req: $\{ts, ck, K_{\text{subsession}}, \ldots\}K_{c,v}\ \{T_{c,v}\}K_{v}$
6. ap_rep: $\{ts\}K_{c,v}$ (optional)

Related work

- Kerberos is based in part on the Needham and Schroeder authentication protocol
  - Authentication Servers
  - Conventional Algorithms
  - Multiple Authentication Servers
- Not including:
  - Public-Key Algorithms
  - Digital Signatures

Related work

- Other approaches for improving Security
  - One-time pass codes: to solve the defect that Kerberos does not protect against the theft of a password through a Trojan horse login program on the user’s workstation
  - Public-key Cryptography: to solve the defect that Kerberos does not support non-repudiation

Results

- Kerberos allows a client to be verified without sending sensitive data through insecure network
  - To authentication server: sending client name, verifier name, expiration time and a random number
  - To verifier: sending a ticket encrypted with the verifier’s secret key ,and current time, a checksum and an optional encryption all encrypted with the session key

Results

- Kerberos allows a client obtain additional tickets by ticket granting service
  - Without cashing user’s password on the workstation
  - Instead, cashing Kerberos ticket and encryption keys only for a short time
  - Within a limited period, ticket granting ticket can help a user to be identified to a new verifier.

Results

- Kerberos 4’s cross-realm authentication allows a user to prove its identity to a new verifier registered in a different realm
  - different authentication servers can share a cross-realm key for a verifier
  - A principal can use ticket granting tick to request a ticket from the new verifier

Results

- Kerberos 5’s multi-hop cross-realm authentication all keys to be shared hierarchically
- MIT reference implementation includes version of popular application such Berkeley R-commands, telnet and POP

Take Away

- Author’s Claim:
  - Show how authentication Service can be implemented to fit in with computer networks
  - Show how passwords can avoid appearing during authentication
  - Show how eavesdropping and replay attack are prevented

Take Away

- Author’s Claim:
  - Show how a service and a user can verify each other’s identity
  - Gives a overall mechanism to expand Kerberos’s usage across organizations

Take Away

- But...
  - Even though the passwords are not presented ,how could you prevent the user’s private key from being stolen in his workstation?
  - How about the shared key between the user and the verifier ?
  - And the attacker eventually can intercept some information useful during “so-called secure authentication”

Take Away

- Thus, do not only depend on only one party but urge users to change passwords or secret keys in regular if necessary