

# Introduction to attacks on protocols

Dr. Arjan Duresi  
Louisiana State University  
Baton Rouge, LA 70810  
Duresi@Csc.LSU.Edu

These slides are available at:

[http://www.csc.lsu.edu/~duresi/CSC7502\\_04/](http://www.csc.lsu.edu/~duresi/CSC7502_04/)



- ❑ Discussion of Voydock and Kent. Examine how some of these principles translated into SSL and IPsec
- ❑ Discussion of Needham and Schroeder paper; Examine the influence of these protocols on Kerberos authentication protocols.

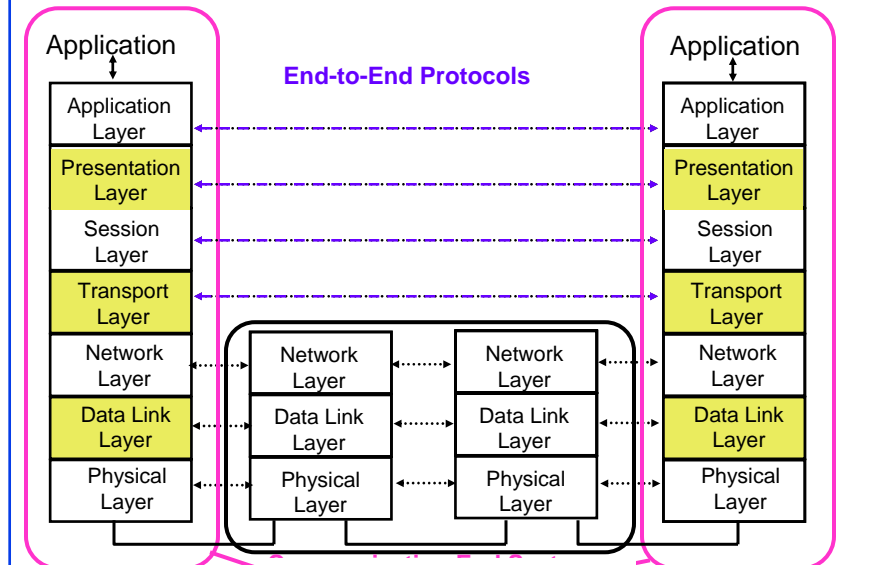
# Transport Layer Security

- ❑ Protocol that allows to establish an end-to-end secure channel, providing: confidentiality, integrity and authentication
- ❑ Defines how the characteristics of the channel are negotiated: key establishment, encryption cipher, authentication mechanism
- ❑ Requires reliable end-to-end protocol, so it runs on top of TCP
- ❑ It can be used by other session protocols (such as HTTPS)
- ❑ Several implementations: for example SSLeay, open source implementation ([www.openssl.org](http://www.openssl.org))

# OSI Reference Model

- ❑ Describes a seven-layer abstract reference model for a network architecture
- ❑ Purpose of the reference model was to provide a framework for the development of protocols
- ❑ OSI also provided a unified view of layers, protocols, and services which is still in use in the development of new protocols
- ❑ Detailed standards were developed for each layer, but most of these are not in use
- ❑ TCP/IP protocols preempted deployment of OSI protocols

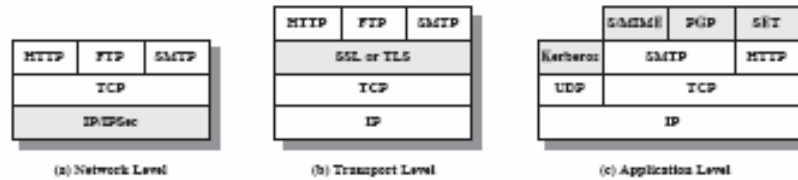
# 7-Layer OSI Reference Model



# Threats

	Threats	Consequences	Countermeasures
<b>Integrity</b>	<ul style="list-style-type: none"> <li>•Modification of user data</li> <li>•Trojan horse browser</li> <li>•Modification of memory</li> <li>•Modification of message traffic in transit</li> </ul>	<ul style="list-style-type: none"> <li>•Loss of information</li> <li>•Compromise of machine</li> <li>•Vulnerability to all other threats</li> </ul>	Cryptographic checksums
<b>Confidentiality</b>	<ul style="list-style-type: none"> <li>•Eavesdropping on the Net</li> <li>•Theft of info from server</li> <li>•Theft of data from client</li> <li>•Info about network configuration</li> <li>•Info about which client talks to server</li> </ul>	<ul style="list-style-type: none"> <li>•Loss of information</li> <li>•Loss of privacy</li> </ul>	Encryption, web proxies
<b>Denial of Service</b>	<ul style="list-style-type: none"> <li>•Killing of user threads</li> <li>•Flooding machine with bogus requests</li> <li>•Filling up disk or memory</li> <li>•Isolating machine by DNS attacks</li> </ul>	<ul style="list-style-type: none"> <li>•Disruptive</li> <li>•Annoying</li> <li>•Prevent user from getting work done</li> </ul>	Difficult to prevent
<b>Authentication</b>	<ul style="list-style-type: none"> <li>•Impersonation of legitimate users</li> <li>•Data forgery</li> </ul>	<ul style="list-style-type: none"> <li>•Misrepresentation of user</li> <li>•Belief that false information is valid</li> </ul>	Cryptographic techniques

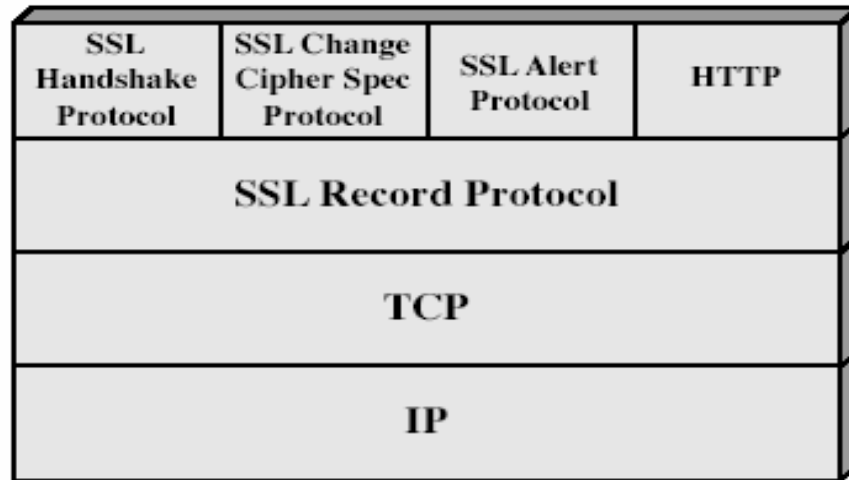
## Location of Security



## SSL (Secure Socket Layer)

- ❑ Transport layer security service
- ❑ originally developed by Netscape
- ❑ version 3 designed with public input
- ❑ subsequently became Internet standard known as TLS (Transport Layer Security)
- ❑ uses TCP to provide a reliable end-to-end service
- ❑ SSL has two layers of protocols
  - SSL record protocol provides basic security services
  - 3 higher-layer protocols:
    - ❑ Handshake, change cipher spec, alert

## SSL Architecture



## SSL Architecture

- **SSL session**
  - an association between client & server
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple SSL connections
  - Once established – operating state
  - During Handshake Protocol – pending read, write states
- **SSL connection**
  - a transient, peer-to-peer, communications link
  - associated with 1 SSL session

## Session and Connection

- ❑ Session parameters:
  - ID, peer certificate, compression method, cipher spec, master secret, is resumable.
- ❑ Connection parameters:
  - Server and client random, server write MAC secret, client write MAC secret, server write key, client write key, IV, sequence number.

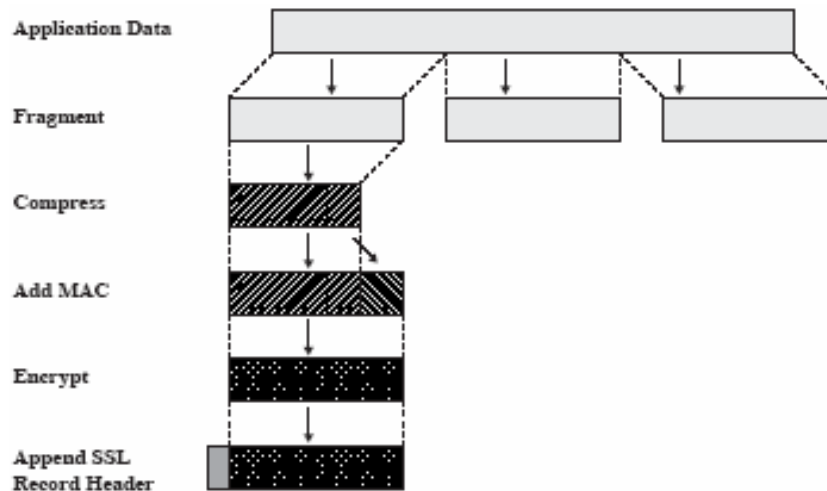
## SSL Record Protocol

- ❑ **Confidentiality**
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - message is compressed before encryption
- ❑ **Message integrity**
  - using a MAC with shared secret key
  - similar to HMAC but with different padding

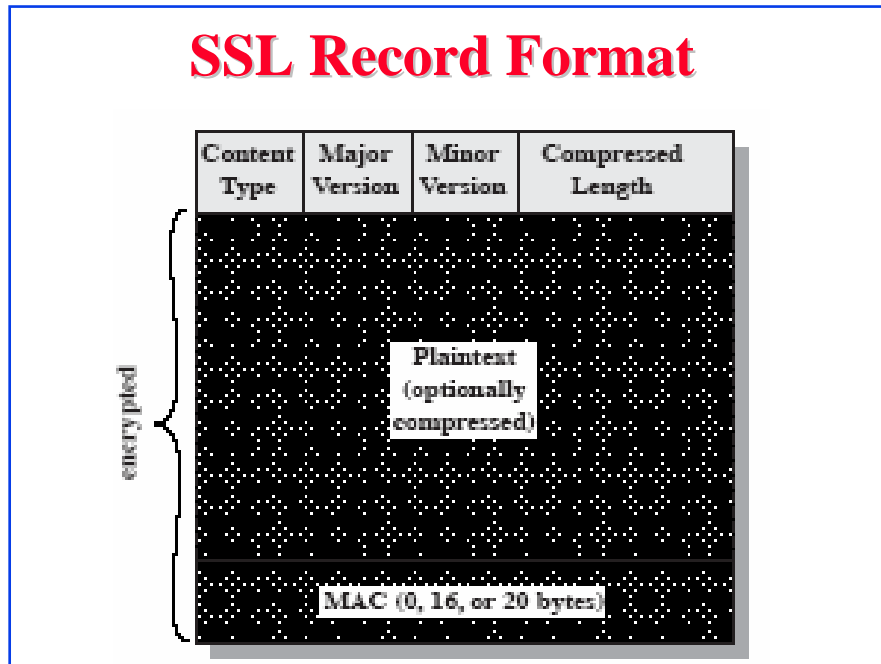
## SSL Record Protocol

- Layered protocol:
  - Fragment application data into blocks
  - Compress data
  - Apply message authentication code (MAC) =  $h(m|s)$  for message  $m$  and secret  $s$
  - Encrypt with client (cw) or server (sw) write key
  - Transmit over TCP
- Specify content type for higher protocols

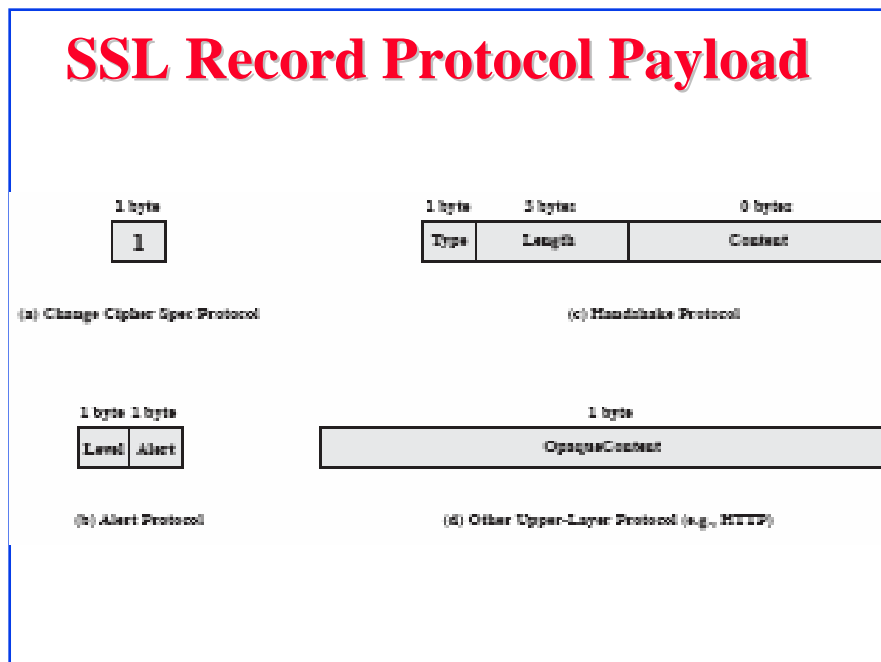
## SSL Record Protocol Operation



# SSL Record Format



# SSL Record Protocol Payload



## SSL Change Cipher Spec Protocol

- ❑ One of 3 SSL specific protocols which use the SSL Record protocol
- ❑ A single message
- ❑ Causes pending state to become current
- ❑ Hence updating the cipher suite in use

## SSL Alert Protocol

- ❑ Conveys SSL-related alerts to peer entity
- ❑ Severity
  - ❑ warning or fatal
- ❑ Specific alert
  - ❑ unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - ❑ close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- ❑ Compressed & encrypted like all SSL data

## SSL Handshake Protocol

- ❑ Allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- ❑ It is used before any application data is transmitted
- ❑ Comprises a series of messages in phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

## Establish Security Capabilities

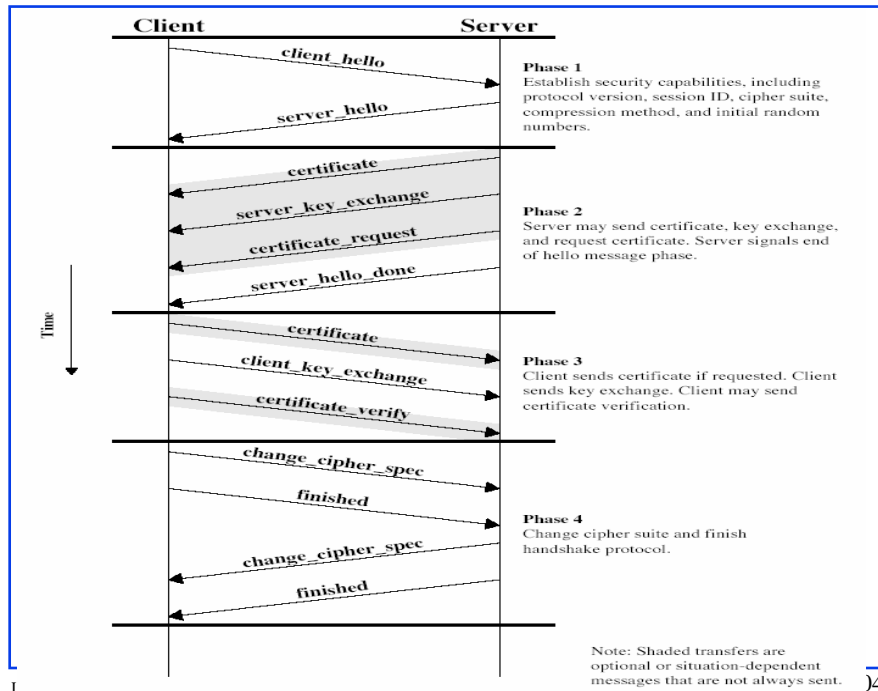
- ❑ Used to initiate a logical connection and to establish the security capabilities associated with it
- ❑ Initiated by client – **client\_hello\_message**, parameters:
  - Version – the highest SSL version understood by the client
  - Random – 32-bit timestamp and 28 bytes random, serve as nonce
  - Session ID – a variable length Session ID. A nonzero value indicates the client wants to update existing connection. Zero – new one
  - CipherSuite – list of combinations of cryptographic algorithms supported by client in decreasing order of preference
  - Compression method – list supported methods

## Establish Security Capabilities

- The server responds with **server\_hello** message with the same parameters as the client\_hello
  - Version
  - Random - independent of client
  - Session ID
  - CipherSuite – the single cipher selected
  - Compression – the method selected

## SSL Handshake Protocol Message Types

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value



## Server Authentication and Key Exchange

- ❑ Server begins this phase by sending its certificates
- ❑ Next sender server sends a **server\_key\_exchange** message (not sent if a certificate with fixed DH was sent or when RSA will be used)
- ❑ Next a nonanonymous server (server not using DH) can request a certificate from the client
- ❑ Final message **server\_done** message

## Client Authentication and Key Exchange

- ❑ Client verifies the certificate of server and the parameters are acceptable
- ❑ If server has requested a certificate, the client sends it with a **certificate** message.
- ❑ Next **client\_key\_exchange\_message**
- ❑ Finally the client sends a **certificate\_verify** message

To provide explicit verification of a client certificate (following any client certificate with signing capability)

## Finish

- ❑ Client send **change\_cipher\_spec**
- ❑ Then **finished message** using the new key

## Cryptographic Computations

- Master secret creation
  - A pre-master-secret is exchanged first.
    - RSA, or Diffie-Hellman.
  - Both sides compute master secret based on pre-master-secret.
- Generation of cryptographic parameters.
  - Client/server write MAC secrets, client/server write keys, client/server write IV are generated from master secret.

## Cryptographic Computations: Details (1)

- Client generates a 48-byte pre-master-secret  $s_p$
- Master secret:
  - $s_m = \text{MD5}(s_p | \text{SHA}('A' | s_p | r_c | r_s)) |$   
 $\text{MD5}(s_p | \text{SHA}('BB' | s_p | r_c | r_s)) |$   
 $\text{MD5}(s_p | \text{SHA}('CCC' | s_p | r_c | r_s))$
  - Where  $r_{c,s}$ : client, server random

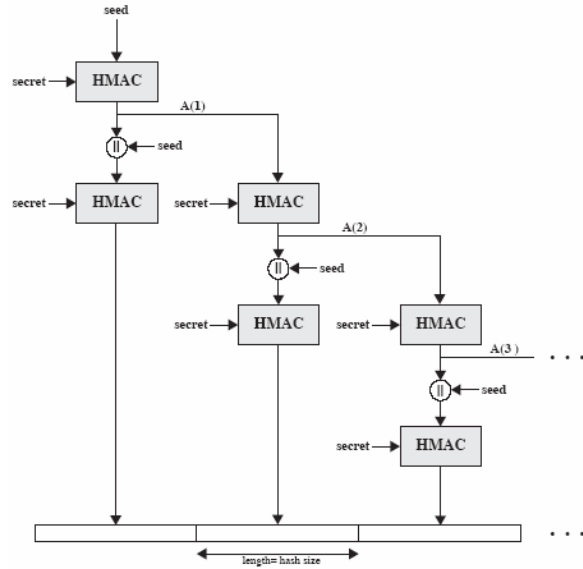
## Cryptographic Computations: Details (2)

- ❑ Session key: same as above, but use the master secret in place of  $s_p$  to generate byte stream to cut out:
  - Client, server MAC secret
  - Client, server write key
  - Client, server IV

## TLS (Transport Layer Security)

- ❑ IETF standard RFC 2246 similar to SSLv3
- ❑ with minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate negotiations
  - changes in use of padding

# TLS Function P\_hash



## Summary



- Have considered:
  - SSL/TLS transport layer security protocols

# IP Security



- ❑ Objectives
- ❑ IPSec architecture & concepts
- ❑ IPSec authentication header
- ❑ IPSec encapsulating security payload

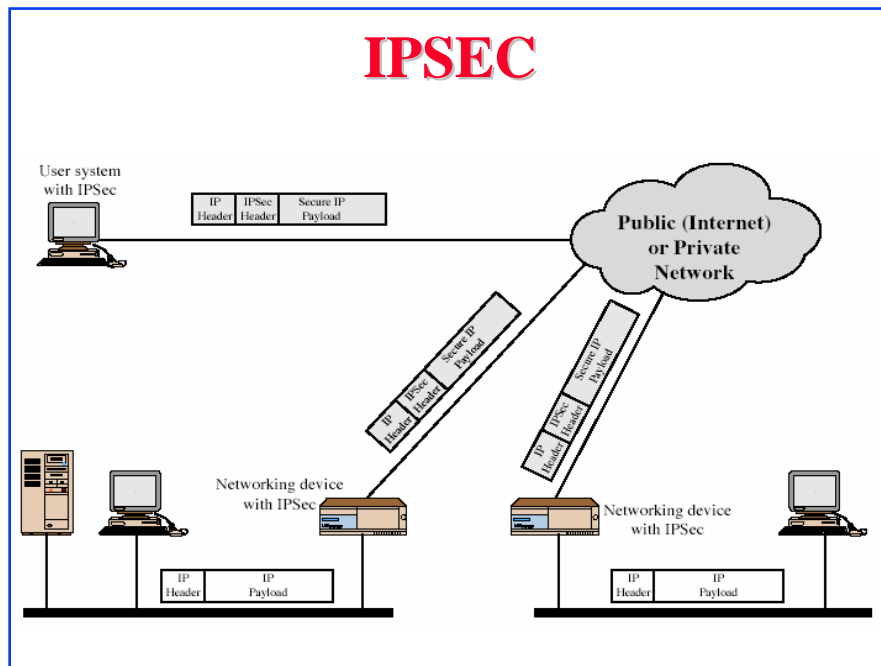
## IPSEC Objectives

- ❑ IPsec – IETF standard for real-time communication security
- ❑ Band-aid for IPv4
  - Spoofing a problem
  - Not designed with security or authentication in mind
- ❑ IP layer mechanism for IPv4 and IPv6
  - Not all applications need to be security aware
- ❑ Can be transparent to users

## IPSEC

- ❑ General IP Security mechanisms
- ❑ Provides
  - authentication
  - confidentiality
  - key management
- ❑ Applicable to use over LANs, across public & private WANs, & for the Internet

# IPSEC



## Benefits of IPsec

- ❑ In a firewall/router provides strong security to all traffic crossing the perimeter
- ❑ Is resistant to bypass
- ❑ Is below transport layer, hence transparent to applications
- ❑ Can be transparent to end users
- ❑ Can provide security for individual users if desired

## IPSec Services

- ❑ Access control
- ❑ Connectionless integrity
- ❑ Data origin authentication
- ❑ Rejection of replayed packets
  - a form of partial sequence integrity
- ❑ Confidentiality (encryption)
- ❑ Limited traffic flow confidentiality

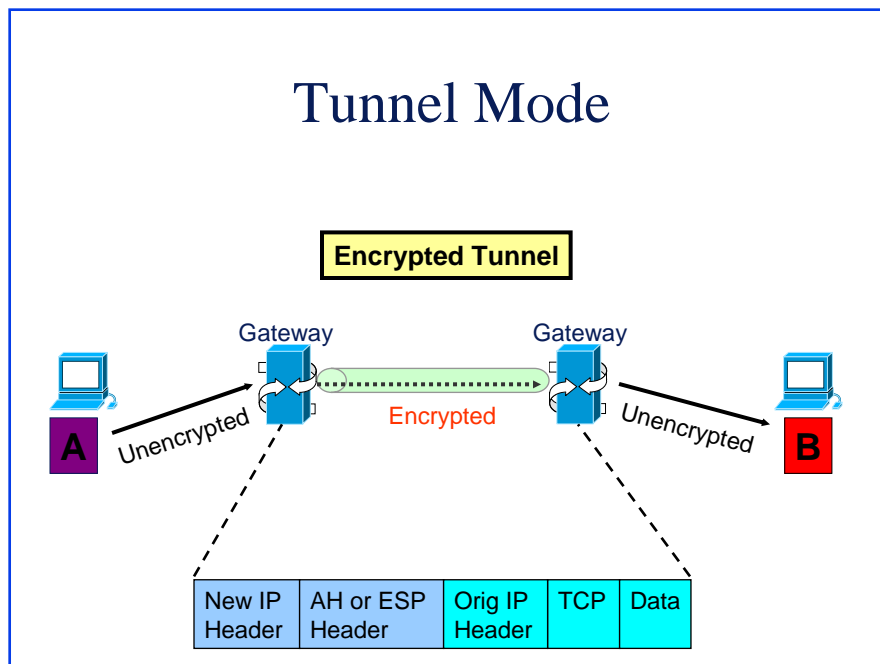
## Architecture & Concepts

- ❑ Specification is quite complex
- ❑ Mandatory in IPv6, optional in IPv4
- ❑ Host or gateway implementation
- ❑ Tunnel vs. Transport mode
- ❑ Security association (SA) – cryptographically protected connection
  - Security parameter index (SPI)
  - Security policy database (SPD)
  - SA database (SAD)
- ❑ Encapsulating security payload (ESP)
- ❑ Authentication header (AH)

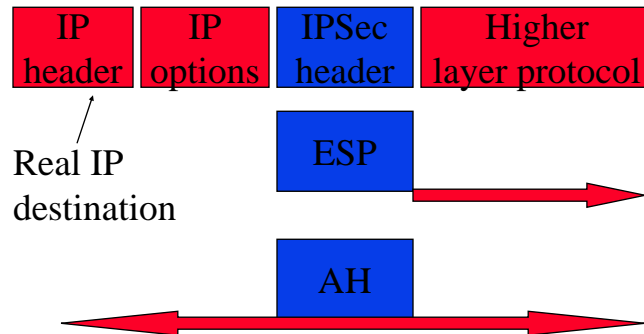
## Hosts & Gateways

- ❑ Hosts can implement IPSec to :
  - Other hosts in transport or tunnel mode
  - Gateways with tunnel mode
- ❑ Gateways to gateways - tunnel mode

## Tunnel Mode

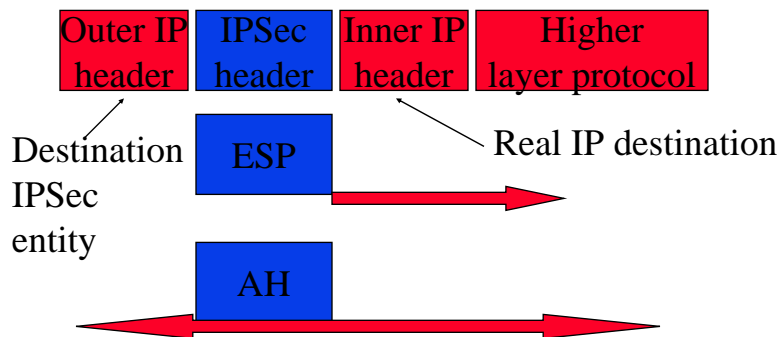


## Transport Mode



- ❑ ESP protects higher layer payload only. Encryption and/or integrity protection
- ❑ AH can protect IP headers as well as higher layer payload
- ❑ Is AH really needed?

## Tunnel Mode



- ❑ ESP applies only to the tunneled packet
- ❑ AH can be applied to portions of the outer header

## Tunnel and Transport Mode

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✓	✓	✓
Connectionless integrity	✓		✓
Data origin authentication	✓		✓
Rejection of replayed packets	✓	✓	✓
Confidentiality		✓	✓
Limited traffic flow confidentiality		✓	✓

## Network Address Translation

- ❑ NAT – share IPv4 addresses. Translates internal IP addresses to globally unique IP addresses
- ❑ IPSec tunnel and transport cannot work with NAT
- ❑ Firewalls vs. End-to-end security

## Security Association - SA

- ❑ A one-way relationship between sender & receiver that affords security for traffic flow
- ❑ Defined by 3 parameters:
  - Security Parameters Index (SPI)
  - IP Destination Address
  - Security Protocol Identifier – indicates AH or ESP assoc.
- ❑ Has a number of other parameters
  - seq no, AH & ESP info, lifetime etc
- ❑ Have a database of Security Associations
- ❑ Determine IPsec processing for senders
- ❑ Determine IPsec decoding for destination
- ❑ SAs are not fixed! Generated and customized per traffic flows

## Security Parameters Index - SPI

- ❑ Can be up to 32 bits large
- ❑ The SPI allows the destination to select the correct SA under which the received packet will be processed (according to the agreement with the sender)
  - The SPI is sent with the packet by the sender
- ❑ SPI + Dest IP address + IPsec Protocol (AH or ESP) uniquely identifies a SA

## SA Database - SAD

- ❑ Holds parameters for each SA
  - Sequence number Counter - 32 bit
  - Lifetime of this SA
  - AH and ESP information
  - Tunnel or transport mode
- ❑ Every host or gateway participating in IPSec has their own SA database

## SA Bundle

- ❑ More than 1 SA can apply to a packet
- ❑ Example: ESP does not authenticate new IP header.  
How to authenticate?
  - Use SA to apply ESP w/out authentication to original packet
  - Use 2<sup>nd</sup> SA to apply AH

## Security Policy Database - SPD

- ❑ What traffic to protect?
- ❑ Has incoming traffic been properly secured?
- ❑ Policy entries define which SA or SA Bundles to use on IP traffic
- ❑ Each host or gateway has their own SPD
- ❑ Index into SPD by Selector fields
  - Dest IP, Source IP, Transport Protocol, IPSec Protocol, Source & Dest Ports, ...

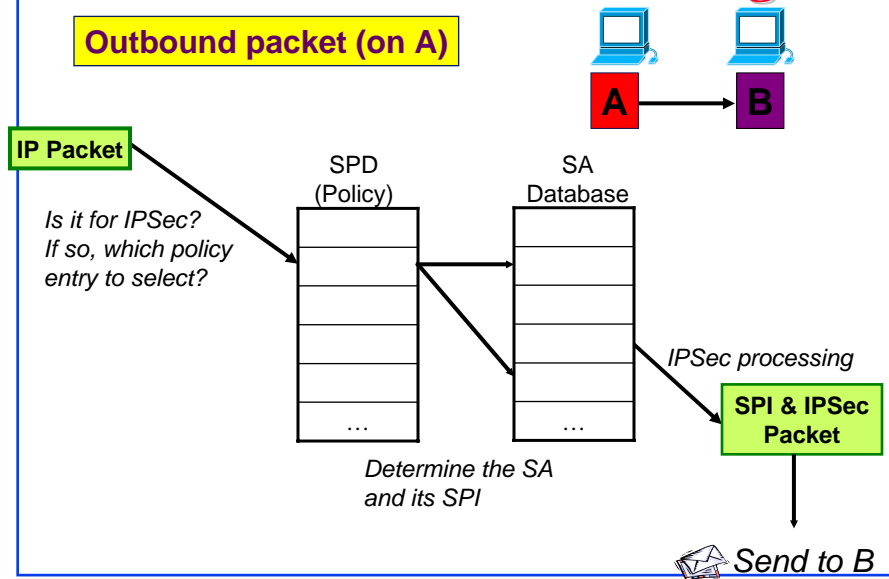
## SPD Entry Actions

- ❑ Discard
  - Do not let in or out
- ❑ Bypass
  - Outbound: do not apply IPSec
  - Inbound: do not expect IPSec
- ❑ Protect – will point to an SA or SA bundle
  - Outbound: apply security
  - Inbound: check that security must have been applied

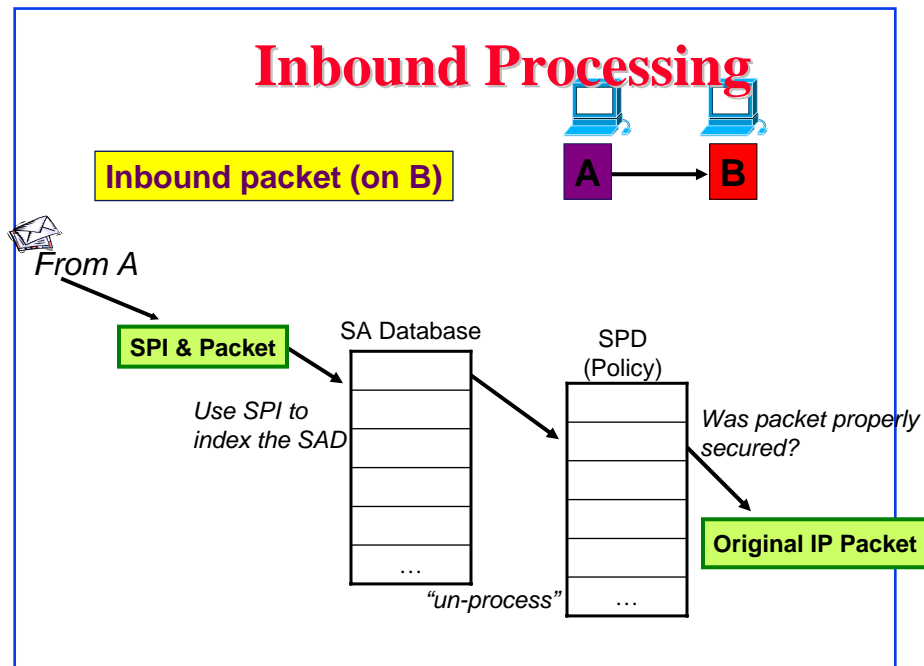
## SPD Protect Action

- If the SA does not exist...
  - Outbound processing: use IKE to generate SA dynamically
  - Inbound processing: drop packet

## Outbound Processing



## Inbound Processing



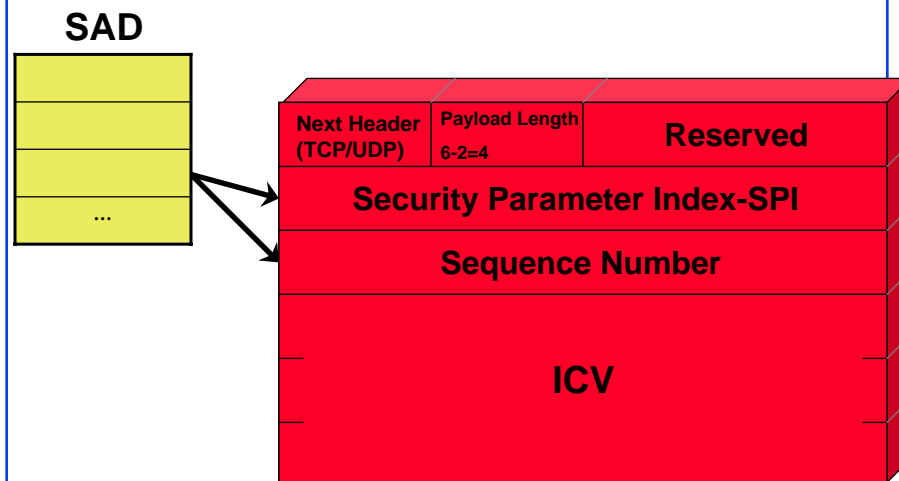
## Authenticated Header

- Data integrity
  - Entire packet has not been tampered with
- Authentication
  - Can “trust” IP address source
  - Use Message Authentication Code (MAC) to authenticate
- Anti-replay feature
- Integrity check value

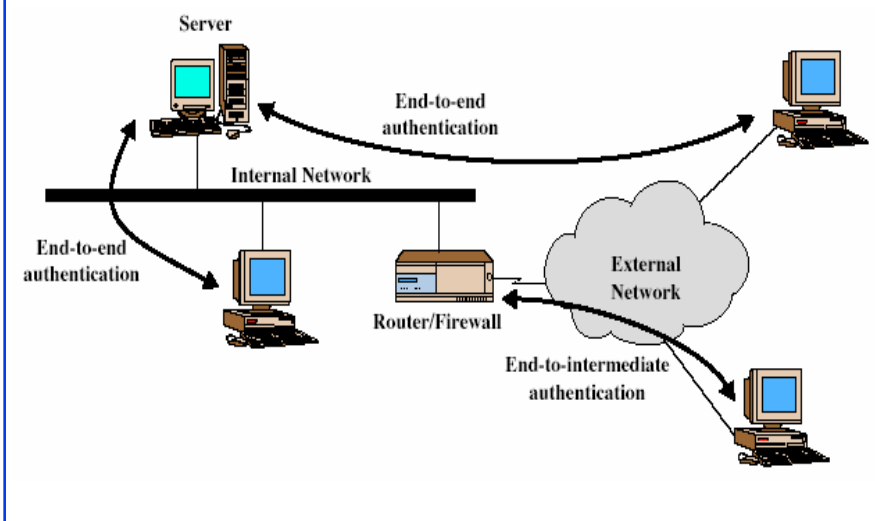
## Integrity Check Value - ICV

- ❑ Message authentication code (MAC) calculated over
  - IP header field that do not change or are predictable such as Source Address, Internet Header Length
  - IPSec protocol header minus where the ICV value goes
  - Upper-level data
- ❑ Code may be truncated to first 96 bits

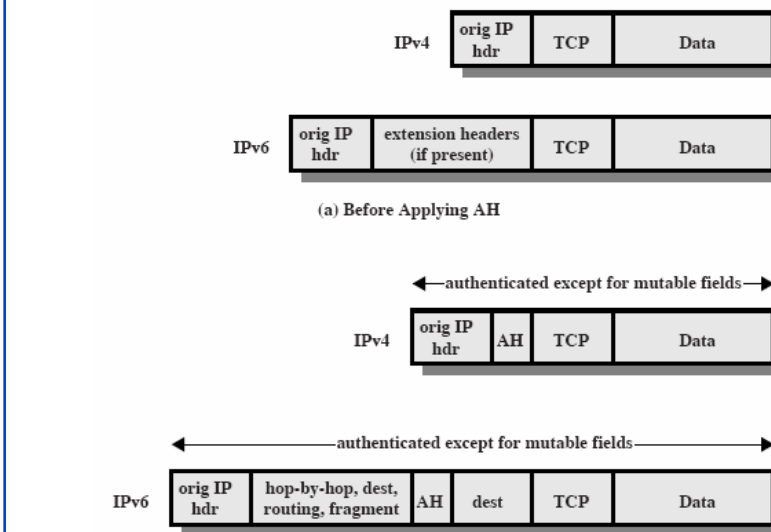
## IPSec Authenticated Header



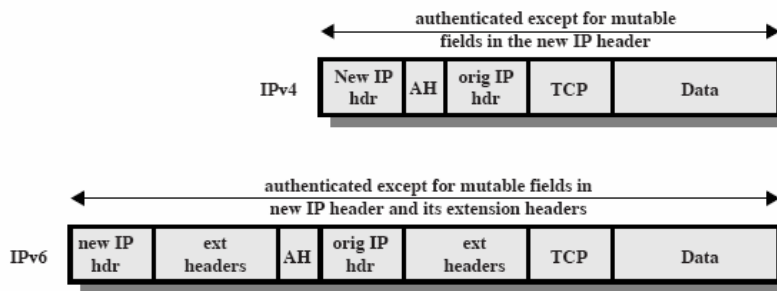
# End-to-end vs. End-to-Intermediate Authentication



# Transport Mode



## Tunnel Mode



## Encapsulated Security Protocol - ESP

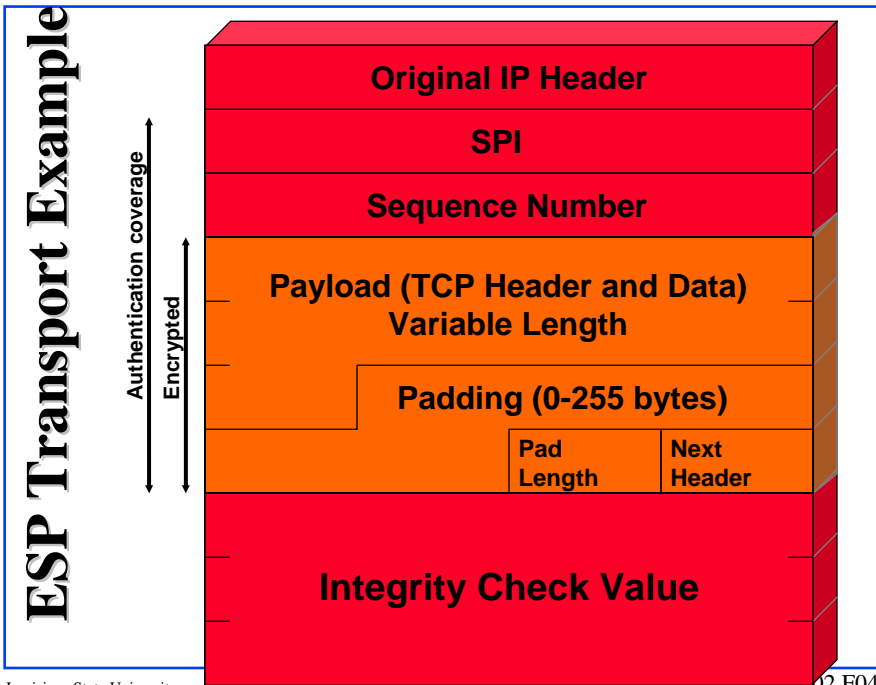
- ❑ Confidentiality for upper layer protocol
- ❑ Traffic flow confidentiality
- ❑ Data origin authentication and connectionless integrity (optional)

## Outbound Packet Processing

- ❑ Form ESP payload
- ❑ Pad as necessary
- ❑ Encrypt result [payload, padding, pad length, next header]
- ❑ Apply authentication
  - Allow rapid detection of replayed/bogus packets
  - Allow potential parallel processing - decryption & verifying authentication code

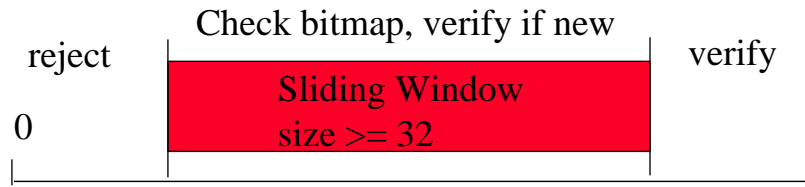
## Outbound Packet Processing...

- ❑ Sequence number generation
  - Increment then use
  - With anti-replay enabled, check for rollover and send only if no rollover
  - With anti-replay disabled, still needs to increment and use but no rollover checking
- ❑ ICV calculation
  - ICV includes whole ESP packet minus *authentication data* field
  - Implicit padding of '0's between *next header* and *authentication data* is used to satisfy block size requirement for ICV algorithm



## Inbound Packet Processing

- Sequence number checking
  - Anti-replay is used only if authentication is selected
  - Sequence number should be the first ESP check on a packet upon looking up an SA
  - Duplicates are rejected!



## Anti-replay Feature

- ❑ Optional
- ❑ Information to enforce held in SA entry
- ❑ Sequence number counter - 32 bit for outgoing IPSec packets
- ❑ Anti-replay window
  - 32-bit
  - Bit-map for detecting replayed packets

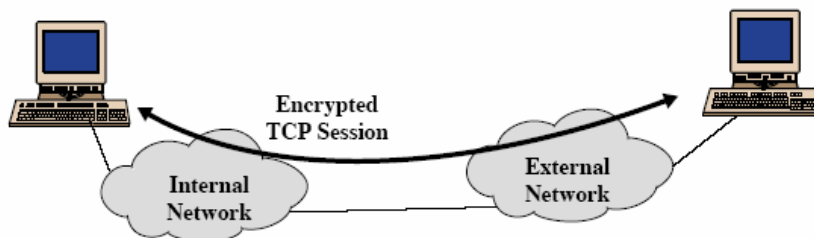
## Anti-replay Sliding Window

- ❑ Window should not be advanced until the packet has been authenticated
- ❑ Without authentication, malicious packets with large sequence numbers can advance window unnecessarily
  - Valid packets would be dropped!

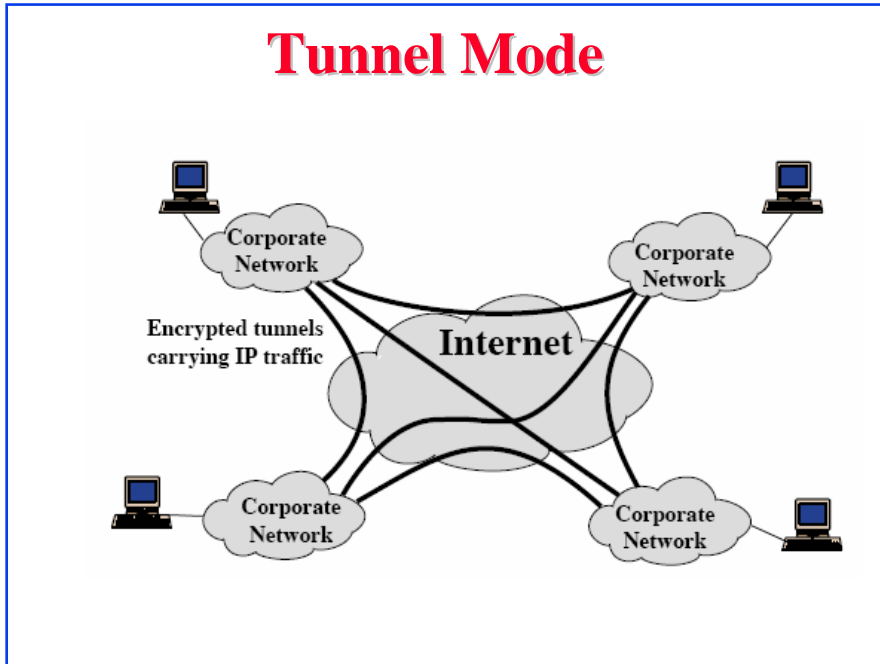
## Inbound Packet Processing...

- ❑ Packet decryption
  - Decrypt quantity [ESP payload, padding, pad length, next header] per SA specification
  - Processing (stripping) padding per encryption algorithm; In case of default padding scheme, the padding field SHOULD be inspected
  - Reconstruct the original IP datagram
- ❑ Authentication verification (option)

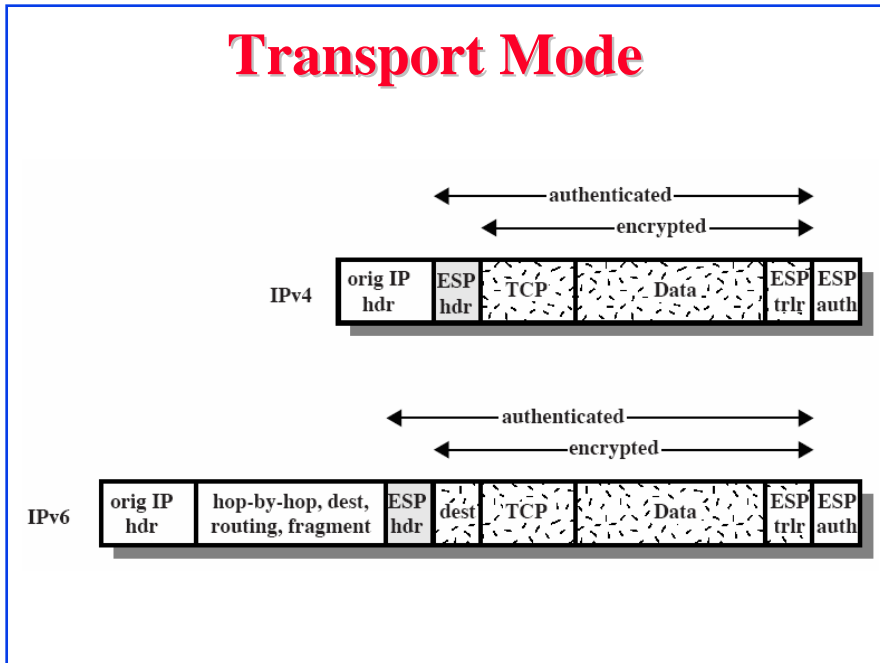
## Transport Mode



# Tunnel Mode



# Transport Mode



## ESP Processing - Header Location...

IPv4

New IP hdr	ESP hdr	Orig IP hdr	TCP	Data	ESP trailer	ESP Auth
------------	---------	-------------	-----	------	-------------	----------

IPv6

New IP hdr	New ext hdr	ESP hdr	Orig IP hdr	Orig ext hdr	TCP	Data	ESP trailer	ESP Auth
------------	-------------	---------	-------------	--------------	-----	------	-------------	----------

- Tunnel mode IPv4 and IPv6

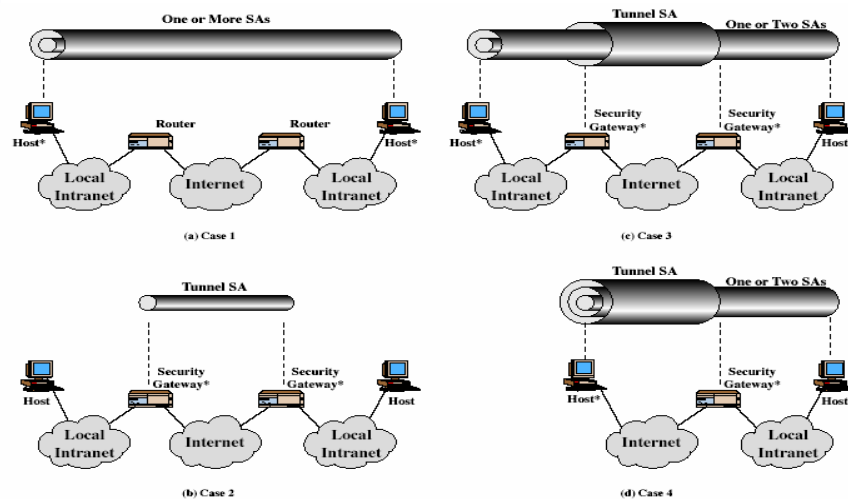
## Transport vs Tunnel Mode ESP

- Transport mode is used to encrypt & optionally authenticate IP data
  - data protected but header left in clear
  - can do traffic analysis but is efficient
  - good for ESP host to host traffic
- Tunnel mode encrypts entire IP packet
  - add new header for next hop
  - good for VPNs, gateway to gateway security

## Combining Security Associations

- ❑ SA's can implement either AH or ESP
- ❑ To implement both need to combine SA's
  - form a security bundle
- ❑ Have 4 cases (see next)

## Combining Security Associations

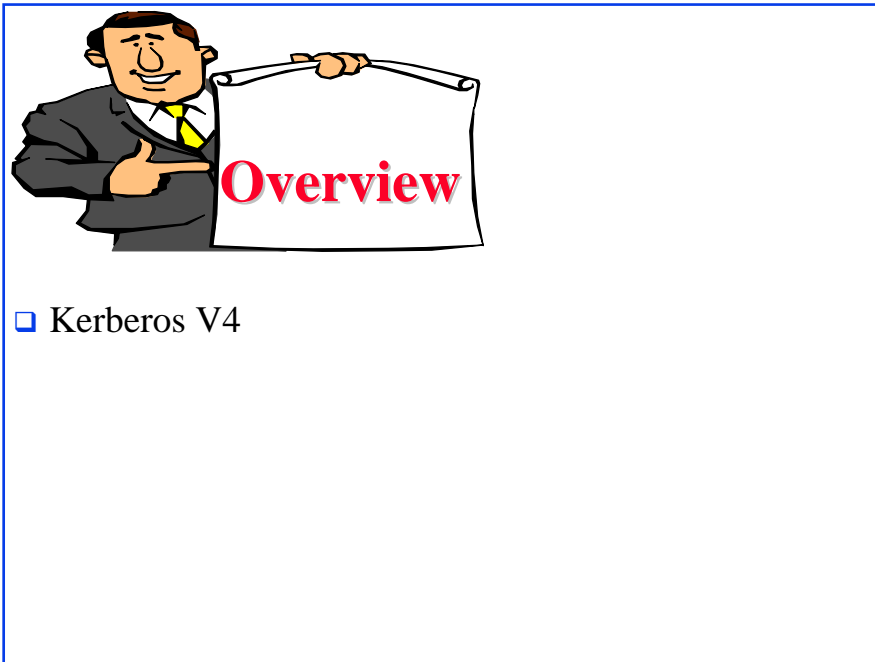


## Summary



- ❑ Objectives
- ❑ IPSec architecture & concepts
- ❑ IPSec authentication header
- ❑ IPSec encapsulating security payload

## Authentication II



- Kerberos V4

## What Is Kerberos?

- Trusted key server system from MIT.  
Recommended reading:  
<http://web.mit.edu/kerberos/www/dialogue.html>
- Provides centralised secret-key third-party authentication in a distributed network
  - allows users secure access to services distributed through network
  - without needing to trust all workstations
  - rather all trust a central authentication server
  - Relieve users/administrators the burden of managing potentially many accounts and passwords

## Kerberos Requirements

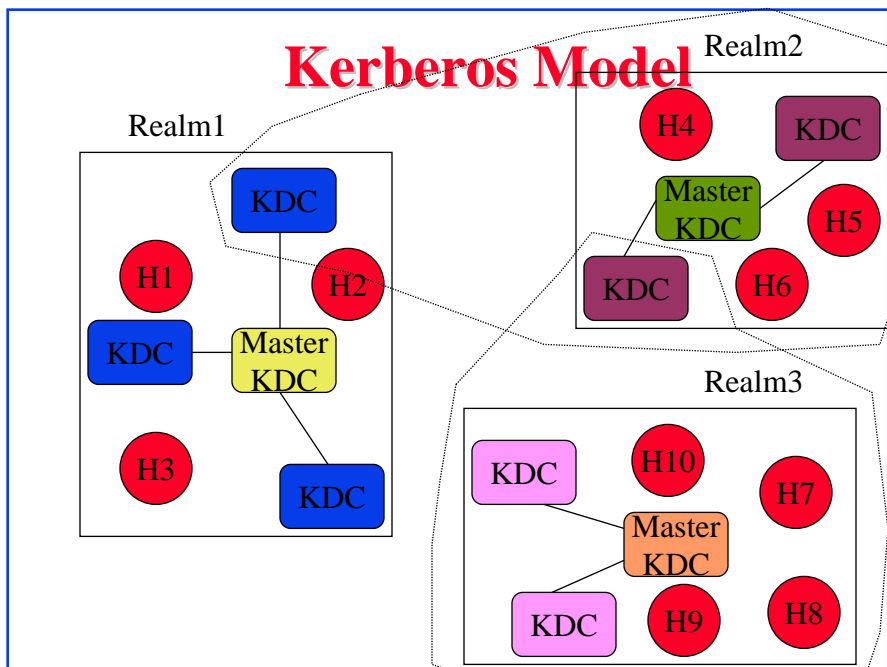
- ❑ First published report identified its requirements as:
  - security
  - reliability
  - transparency
  - scalability
- ❑ Implemented using an authentication protocol based on Needham-Schroeder

## Kerberos

- ❑ Two versions 4 and 5
- ❑ Version 4
  - has a greater installed base
  - Simpler
  - Better performance
  - Works only on TCP/IP
- ❑ Version 5 has greater functionalities

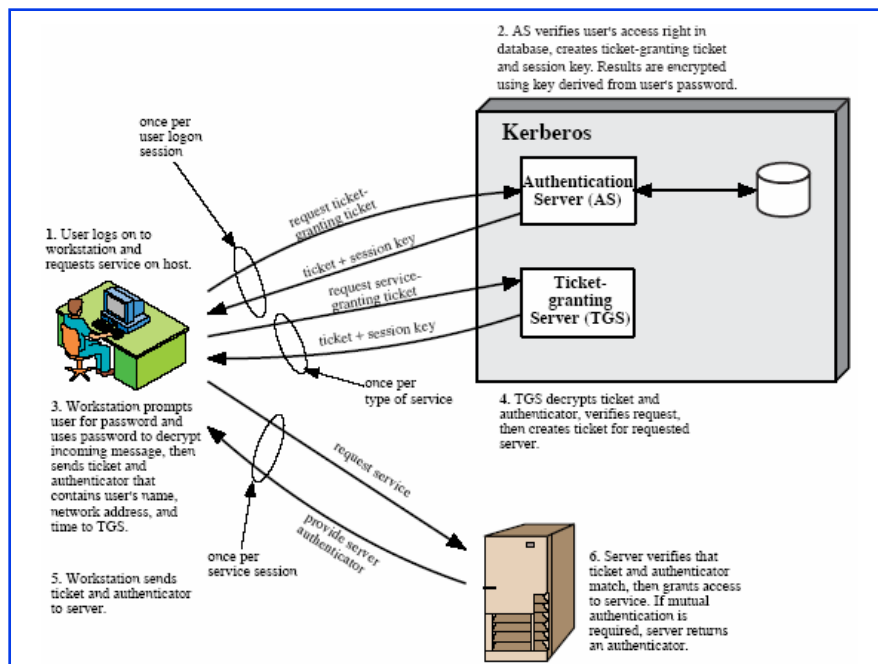
## Kerberos 4 Overview

- ❑ 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 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 Deployment...

- ❑ KDCs are “physically” secured
- ❑ Kerberos libraries are distributed on all nodes with users, applications, and other Kerberos-controlled resources
- ❑ All Kerberos exchanges are protected against confidentiality and integrity attacks
- ❑ Kerberos-rized applications
  - telnet
  - rtools (rlogin, rcp, rsh)
  - Network file systems (NFS/AFS)

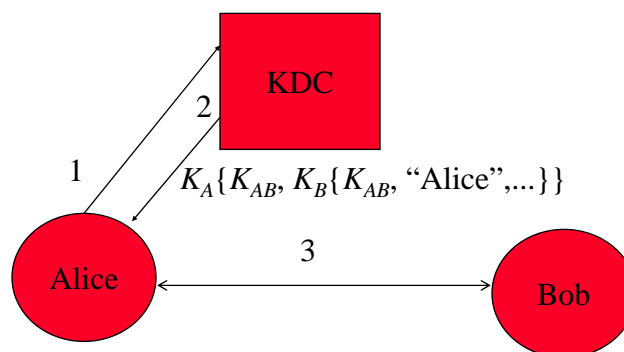
## Where To Start...

- ❑ Every principal has a master (secret) key
  - Human user’s master key is derived from password using DES
  - Other resources must have their keys configured in
- ❑ Every principal is registered with the Kerberos server, i.e. KDC
- ❑ All principals’ master keys are stored in the KDC database, encrypted using the KDC master key

## Tickets ...

- ❑ Every principal has a main shared secret with the KDC - principal's master key
- ❑ Any secure communication/access among principals must be "mediated" by KDC through *tickets*
- ❑ How would Alice talk to Bob?

## Alice, Bob, and KDC



Ticket to Bob:  $K_B\{K_{AB}, \text{"Alice"}, \dots\}$

## Session Key and Ticket-granting Ticket (TGT)

- ❑ Messages between a host and the KDC can be protected using the principal's master key
- ❑ For every request to KDC from the principal:
  - Insists on principal retyping in the password
  - Remember the principal's password
  - Remember the principal's master key derived from the password
- ❑ All options are equally inadequate!

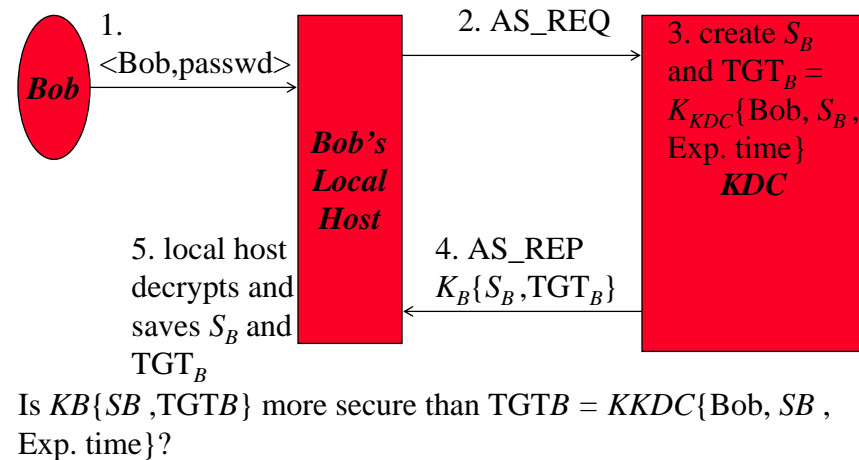
## Session Key and TGT...

- ❑ To avoid potentially too much exposure to password/master key
  - At initial login, a per principal session key  $S_B$  (for Bob) is requested from KDC
  - $S_B$  has a limited valid time period
  - A TGT for Bob is also issued by the KDC, which includes the session key  $S_B$  and Bob's identification information, all encrypted using the KDC's master key

## Session Key and TGT...

- ❑ Bob's Kerberos client (e.g., the login host) decrypts and remembers:
  - $S_B$ , for subsequent message with KDC
  - TGT, for reminding/convincing KDC to use  $S_B$  with it as well
  - No need for remembering/storing password
- ❑ New request to KDC must include TGT in the request message
- ❑ New tickets from KDC must be decrypted with  $S_B$

## Login



## Login

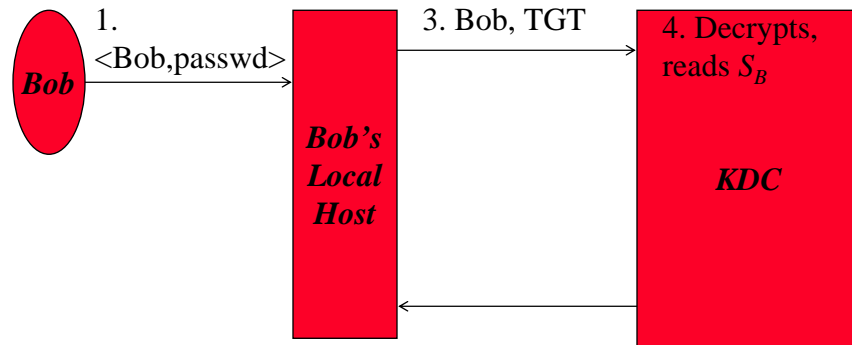
- ❑ Kerberos V4 does not prompt the user for the password until after the workstation has received the credentials from KDC
  - To keep the password for minimum time
- ❑ Kerberos V5 has the user type the password before sending the credentials
  - Make less easy to obtain information which could be used for off-line guessing
- ❑ TGT enables KDC to operate without having to remember state of operations
  - For each request, KDC sends a response and forget about it

## Login

- ❑ What if the workstation generates the TGT?

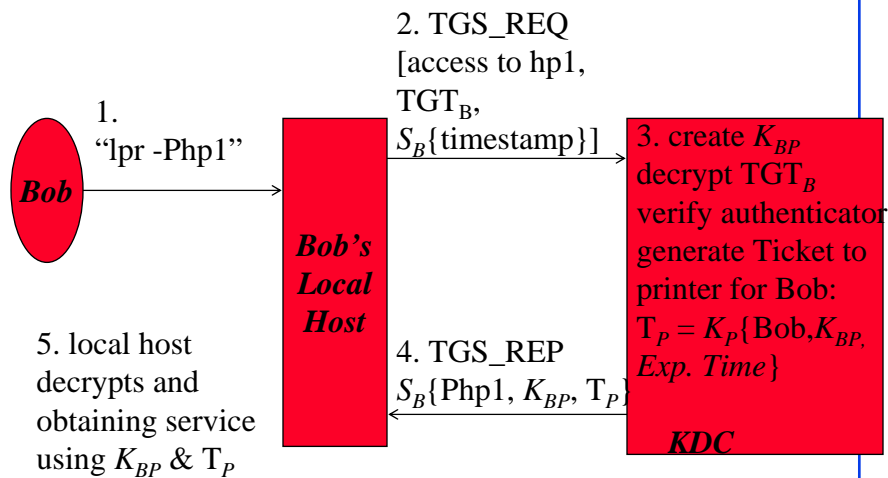
# Login

2 . Bob host generates TGT =  $K_B\{\text{Bob}, S_B\}$



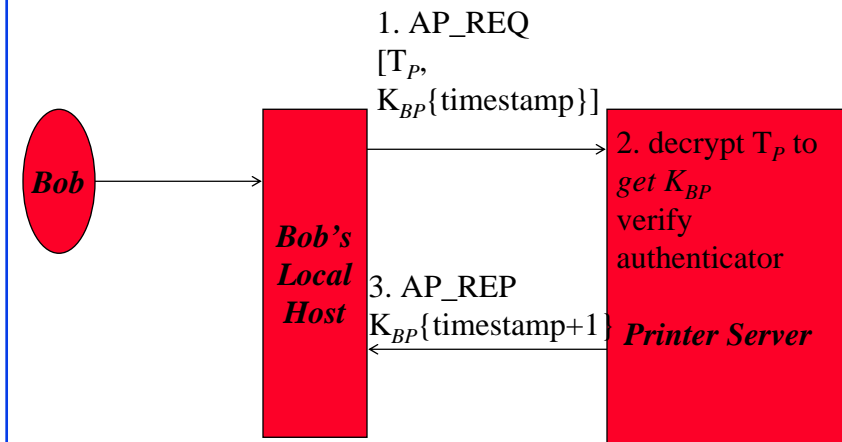
If Bob changes password and  $K_B$  - invalidates TGT

# Need A Ticket...



Why are TGS and AS the same thing - KDC?

## Accessing the Printer...



## Authentication and Global Clock Synchronization

- ❑ Authenticator ==  $K_X\{\text{timestamp}\}$
- ❑ Global clock sync is implied
- ❑ Is the authenticator for TGS\_REQ necessary?
  - The TGS\_REP is encrypted with  $S_B$
- ❑ What about the AP\_REQ?
- ❑ Main purposes of authenticator is to avoid
  - replay of old requests to the same server
  - replay of request on one server to another (server farm, shared principal's master key)

## Kerberos

- ❑ Kerberos could be used for:
  - Authentication
  - Integrity-protected
  - Encryption and integrity-protected
- ❑ The decision is a trade-off between performance vs. security

## Replicated KDCs

- ❑ A single KDC – a single point of failure
- ❑ Multiple replica of KDC - availability and performance. The KDC are interchangeable, share the same master key and database
- ❑ Keeping KDC databases consistent
  - Single master KDC as the point of direct update to principals' database entries
  - Updated database is downloaded from the master to all replica KDCs
  - Periodic download or on-demand

## Will It Be Effective?

- ❑ KDC dynamic state consists of outstanding TGTs and tickets.
- ❑ Kerberos puts the burden of “maintaining” them on the clients - hosts/servers/grantees.
  - Convince me that I did this for you...
- ❑ KDC is only involved in the initial “mediation” and it stays “out of the picture” once a ticket is issued.
- ❑ Only static state information is principals’ database – **read-only** for all replica KDCs.

## Database Content Protection

- ❑ Encryption is required for sensitive data
- ❑ Integrity of the database must be ensured
  - Installation of masqueraded master keys
  - Substitution (replay) of old databases
- ❑ Kerberos stores principals’ master keys encrypted with KDC master key
- ❑ Kerberos transmits a secure hash of the database with encryption, in a separate message during downloads
- ❑ The master key for human users is derived from their password. Other resources are configured with their own master keys

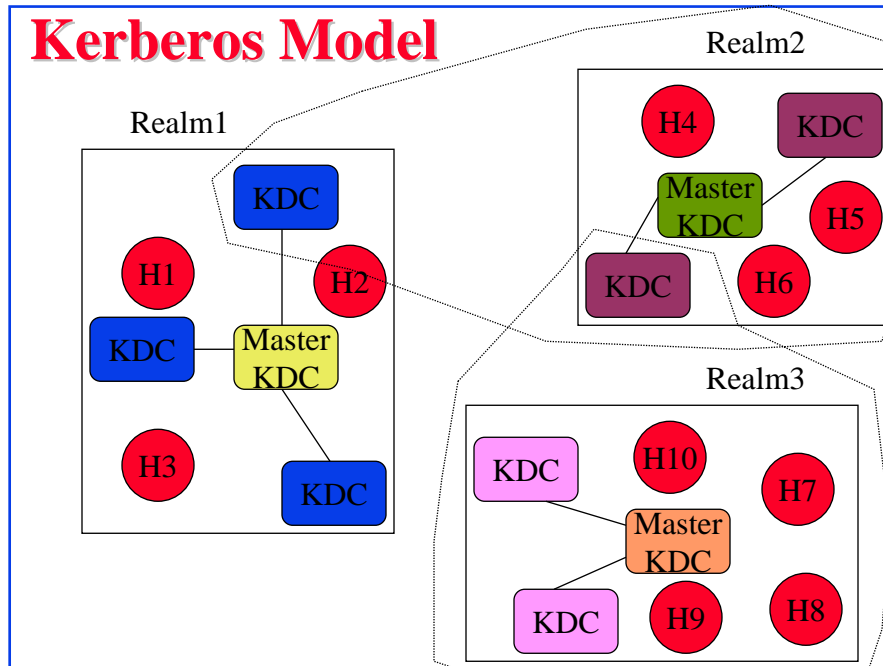
## Multiple Trust Domains

- ❑ Single master KDC can only stretch so far...
- ❑ KDC asks people to put too much trust in it.
  - Should competing commercial entities use the same KDC?
  - .gov, .org, .edu etc, each having a different model of “what is more trustworthy.”
- ❑ Single master KDC - greatest temptation - biggest security risk/vulnerability.
- ❑ So comes different domains or *realms*.

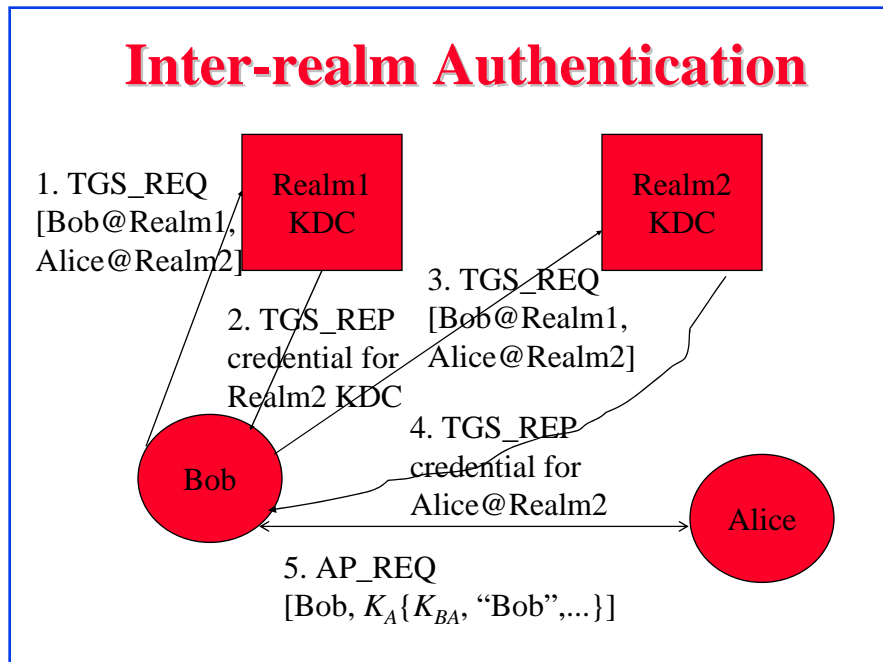
## Kerberos Realms

- ❑ Each realm has a different master KDC, with different master KDC key
- ❑ Each realm can have many replica KDCs, but all sharing the same KDC master key
- ❑ Two KDCs in different realms have different principals' master key databases

# Kerberos Model

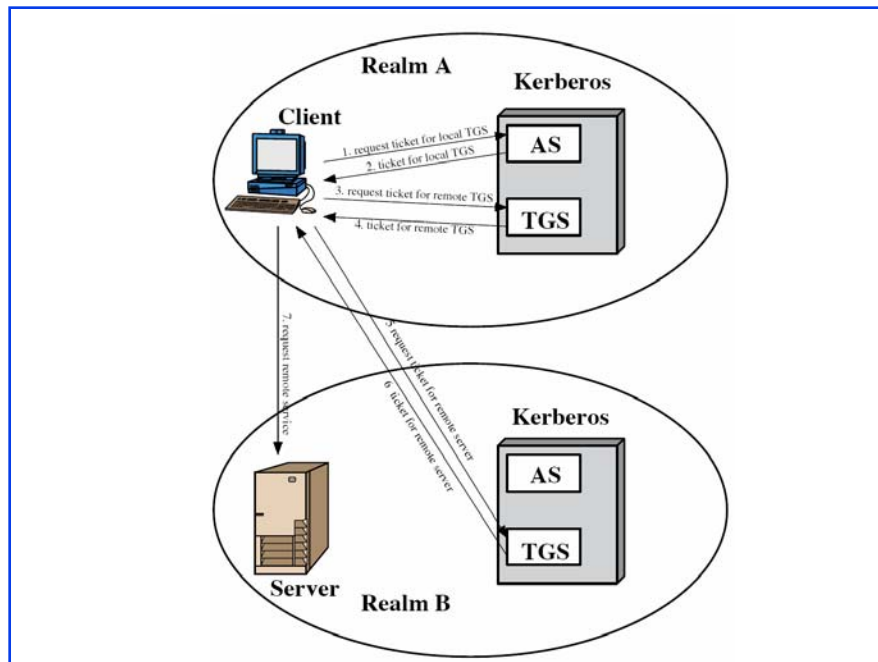


# Inter-realm Authentication



## Inter-realm Authentication

- ❑ Kerberos deliberately prevents access through a chain of KDCs to avoid a rogue KDC to impersonate users from other realms
- ❑ Realms A and B share a key and realms B and C share another key
- ❑ Alice@A would like to talk to Carol@C
  - Alice can get a ticket to B, then request and get from B a ticket to C
  - When Alice ask C for a ticket to Carol the TGS\_REQ will have realm field B and in the ticket Alice's realm A. C will refuse to issue a ticket for Carole because the two realms don't match



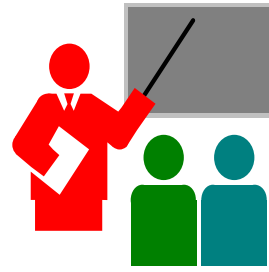
## Kerberos Version 5

- ❑ 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

## Security of Kerberos

- ❑ It may be possible to cache and replay old authenticators. Servers might not be able to keep all tickets
- ❑ If a host can be fooled about the correct time, then an old authenticator can be used. Most network time protocols are insecure. This can be a serious problem
- ❑ Kerberos is vulnerable to password-guessing. An intruder can collect tickets and then try to decrypt them
- ❑ The most serious attack – malicious software . If Kerberos client is replaced with a version that records passwords...
- ❑ Kerberos is not in the public domain. The MIT code is freely available

## Summary



- ❑ Kerberos trusted key server system

## Thank You!

