

# Security Protocols

## Lecture 1

Catalin Leordeanu

[catalin.leordeanu@upb.ro](mailto:catalin.leordeanu@upb.ro)

# Team

\* **Catalin Leordeanu** ([catalin.leordeanu@upb.ro](mailto:catalin.leordeanu@upb.ro))

- The guy talking right now



\* **Elena Apostol** ([elena.apostol@upb.ro](mailto:elena.apostol@upb.ro))

- Will give lectures later



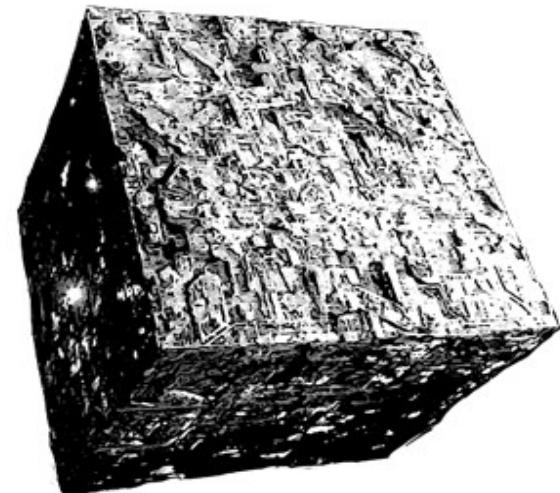
\* **Razvan Popa** ([razvan.popa@stud.acs.upb.ro](mailto:razvan.popa@stud.acs.upb.ro))

- In charge of labs



# Course organization

- **Lectures** 2h/week
  - Classic lectures with slides
- **Topics:**
  - ❖ Access control
  - ❖ SSL/TLS
  - ❖ Internet security
  - ❖ Wireless security
  - ❖ IDS/IPS
  - ❖ ...
- **Practical works (lab)**
- **Essay/Homework** (an essay which will be uploaded on Moodle)
- **2 Quizzes**



# Grading

- Lab: 2 points
- Homework: 3 points
- Short quizzes: 1 point
- Final exam : 4 points

**TOTAL: 10 points**

---

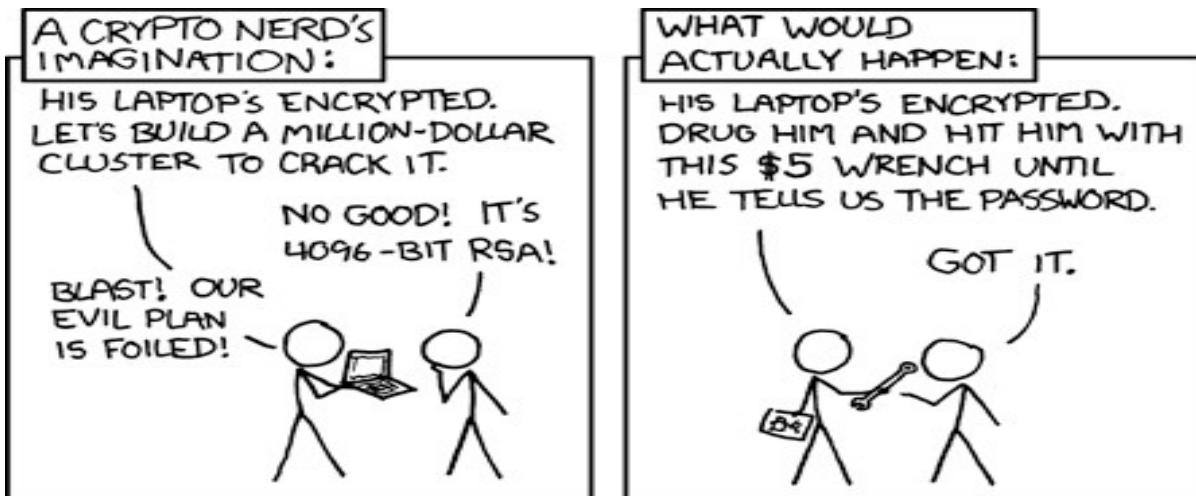
- Quiz 1: week 7, 0.5 points
- Quiz 2: week 14, 0.5 points



Try not.  
Do, Or do not.  
there is no try.

-Yoda

# Basic security concepts



# Policies and mechanisms

## Security policy:

- a specification of what is allowed and what is not from a security point of view

## Security mechanism:

- A method or a tool used to enforce a security policy.
- Typical mechanisms: **encryption, authentication, authorization, audit, etc.**

# Security – basic concepts: Confidentiality, integrity, availability (CIA)

## Confidentiality:

- Limit information access
- Allow access only to authorized users

## Integrity:

- Preserve a degree of trust for data/resources

## Availability:

- Ability to use desired data/resources

# Attacks

## Types of attacks:

### Interception

Unauthorized access to information



### Modification (or alteration)

Unauthorized modification of information

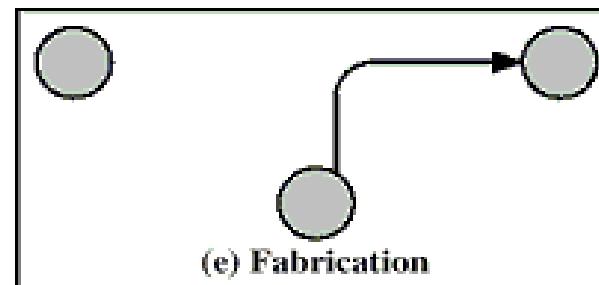
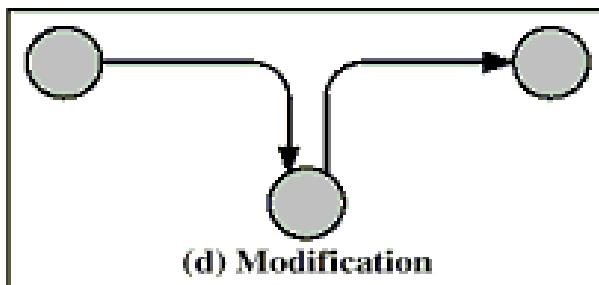
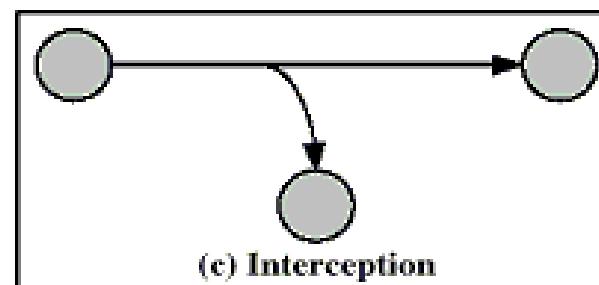
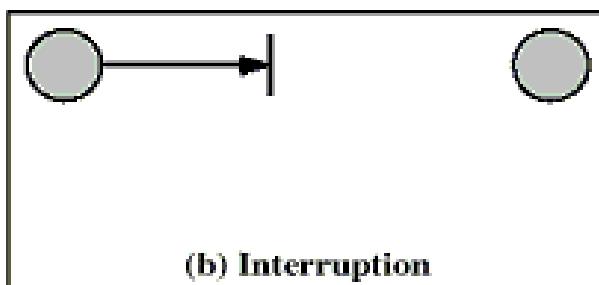
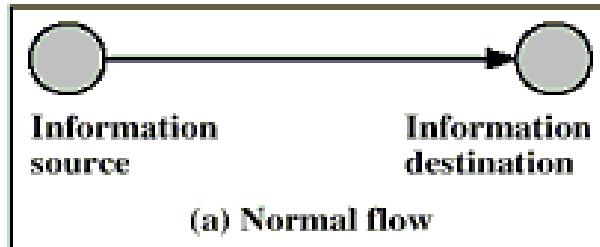
### Fabrication

Creation of false data, fraud

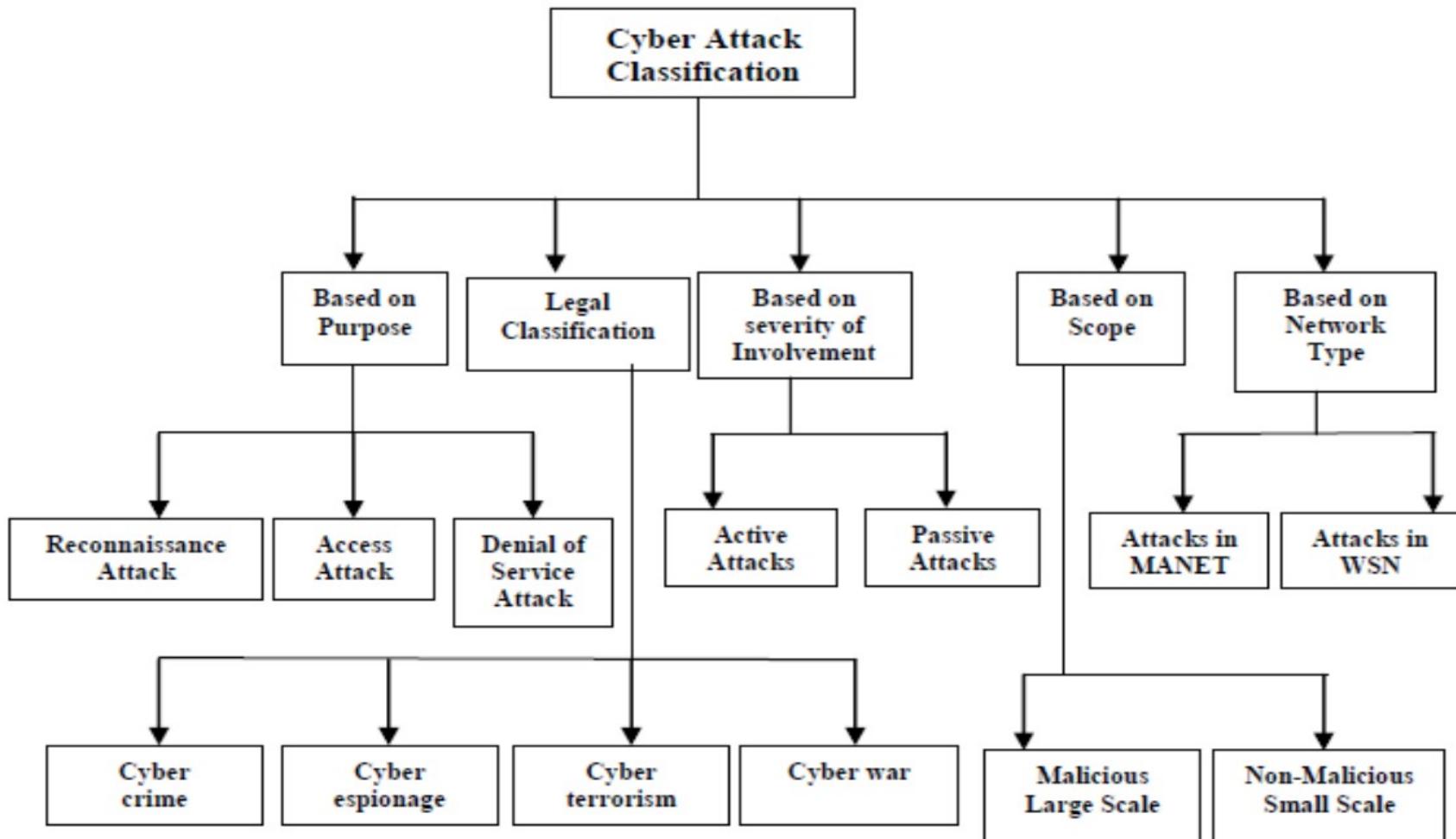
### Interruption of functionality

The interruption or prevention of correct functionality for a given service

# Types of Threats



# Classification of attacks



# Attacks Based on Purpose

## ➤ Reconnaissance Attack

- Packet Sniffers
- Port Scanning
- Ping Sweeping

## ➤ Access Attack

- Attacks on Secret Code
- Utilization of Trust Port
- Port Redirection
- Man-in-the-middle Attacks
- Social Engineering
- Phishing

## ➤ Denial of service Attack

- Host Based, Network Based, Distributed

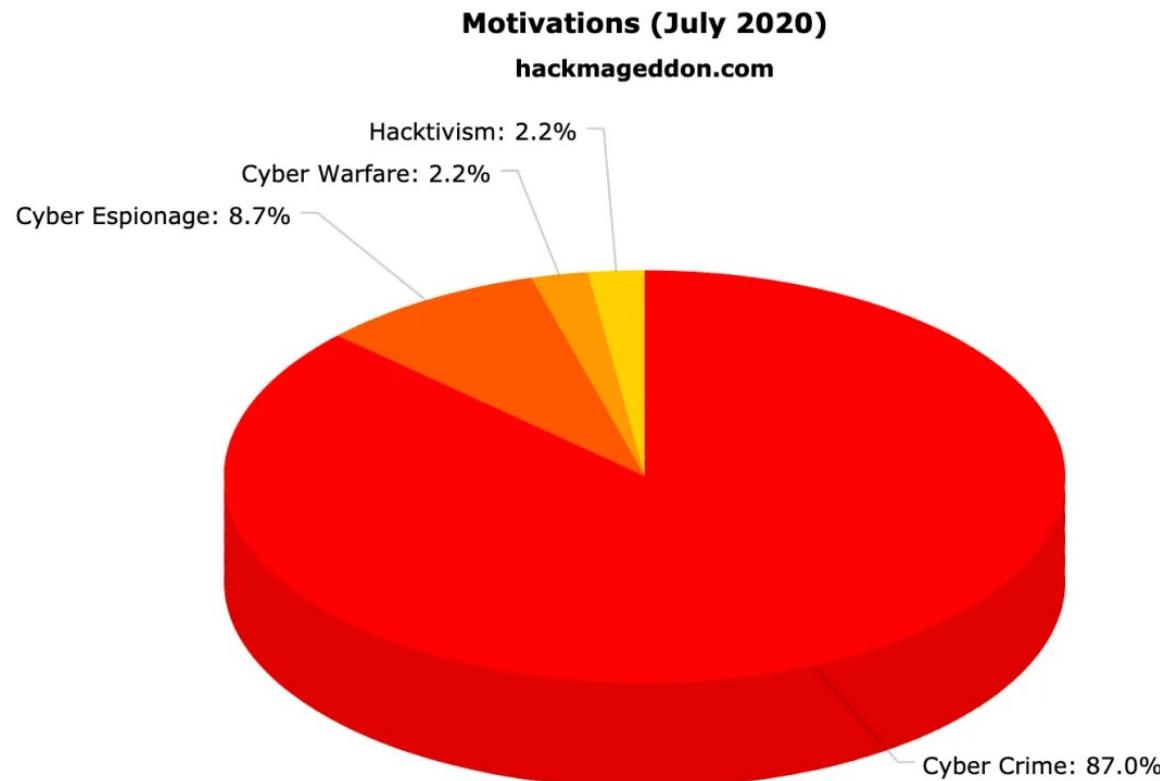
# Legal Classification

- **Cyber crime**
  - Identity theft
  - Credit card fraud
- **Cyber espionage**
  - Tracking cookies
  - RAT controllable
- **Cyber terrorism**
  - Crashing the power grids via a network
  - Attacks on public water system
- **Cyberwar**
  - Russia's war on Estonia (2007)
  - Russia's war on Georgia (2008)

# Based on Severity of Involvement

- **Active Attacks**
  - An attack with data transmission to all parties thereby acting as a liaison enabling severe compromise
    - Masquerade
    - Reply
    - Modification of message
- **Passive Attacks**
  - An attack which is primarily eaves dropping without meddling with the database
    - Traffic analysis
    - Release of message contents

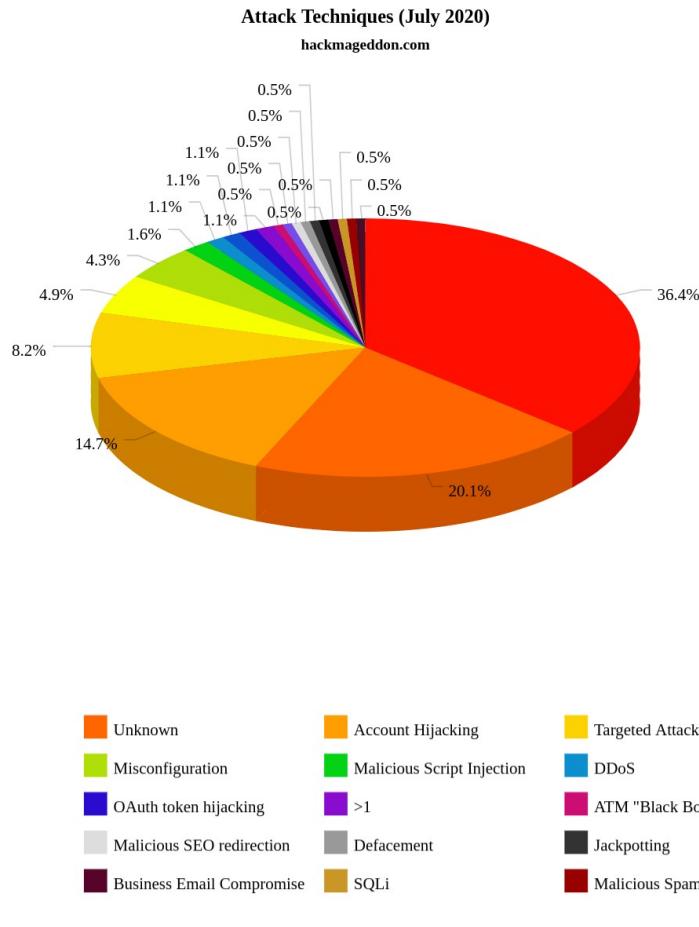
# Motivation of attacks



Source: <http://www.hackmageddon.com/category/security/cyber-attacks-statistics/>

# Attack techniques

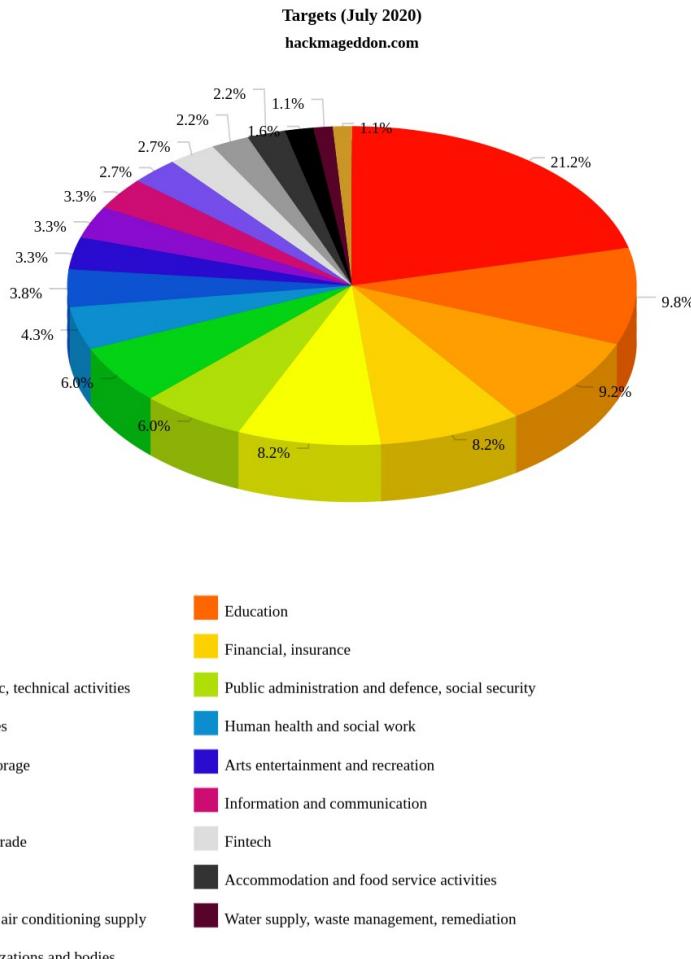
JS chart by amCharts



Source: <http://www.hackmageddon.com/category/security/cyber-attacks-statistics/>

# Attack targets

JS chart by amCharts



Source: <http://www.hackmageddon.com/category/security/cyber-attacks-statistics/>

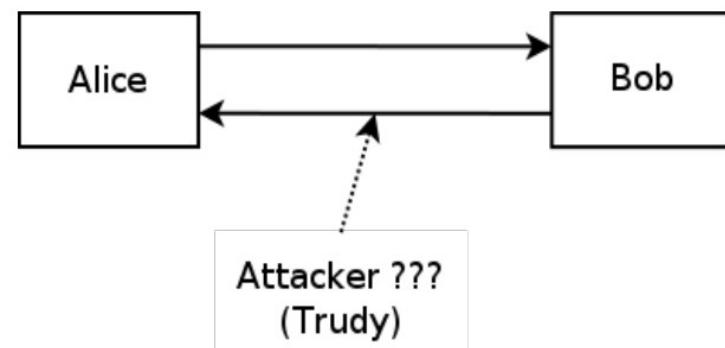
# Cryptology– basic concepts

A mechanism used to ensure security for data transfer or data storage. Prevents third parties from reading private messages.

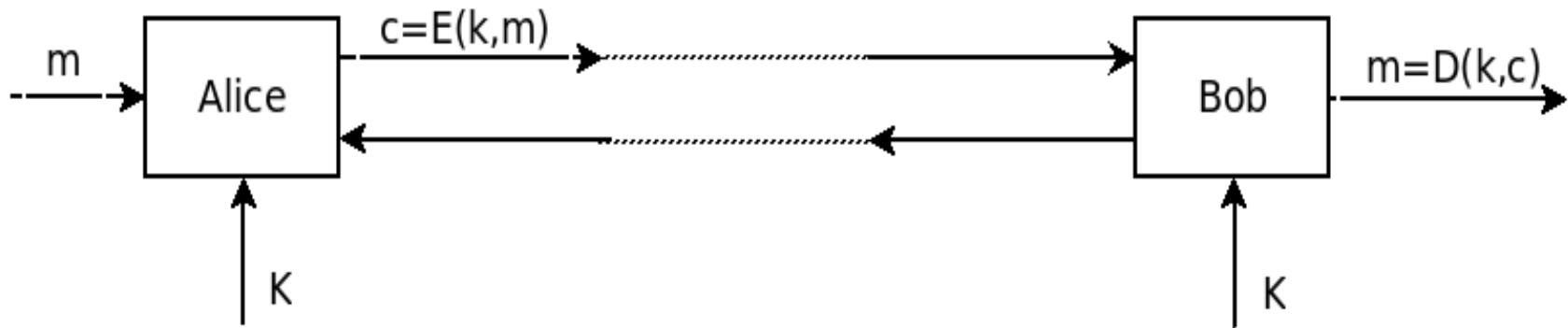
**Cryptology = Cryptography + Cryptanalysis**

Utilizare:

- Data encryption, protection against data leakage
- Anonymous communication
- Ensures data integrity (Hash functions)
- Verify data origin (digital signatures)
- Verify identity (X.509 certificates)
- *Everything else...*



# Symmetric encryption



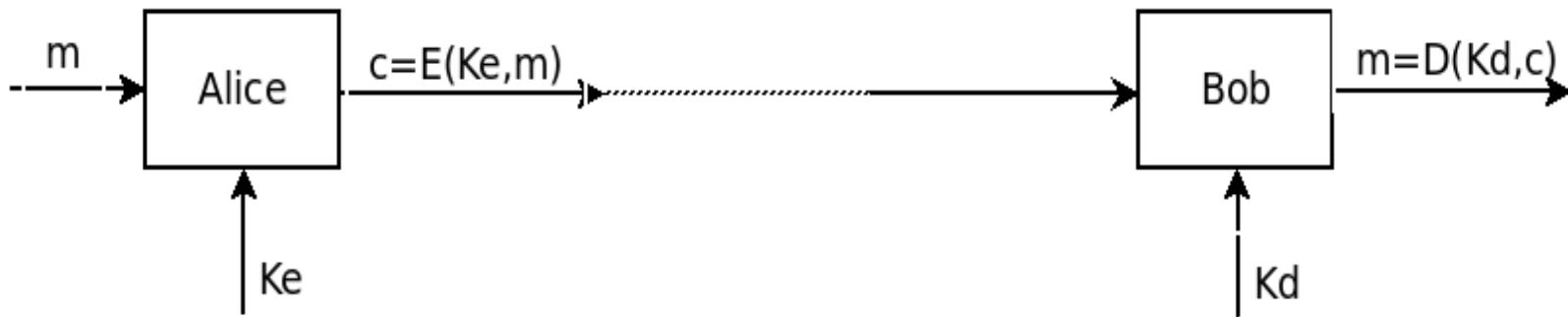
**Uses the same key K for encryption and for decryption**

- $m$  – clear text
- $c$  – ciphertext
- Encryption:  $c=E(k,m)$
- Decryption:  $m=D(k,c)$

$$m=D(k,E(k,m))$$

**Vulnerability:** How do Alice and Bob get to know key k?

# Asymmetric encryption



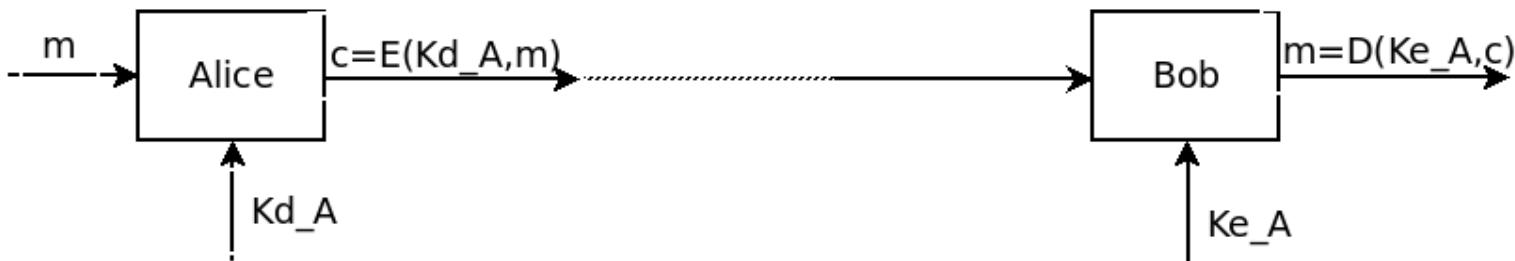
**Uses a public key for encryption ( $K_e$ ) and a private key for decryption( $K_d$ ), known only by Bob.**

- $K_e$  – encryption key (public)
- $K_d$  – decryption key (private)
- Encryption:  $c=E(k_e,m)$
- Decryption:  $m=D(k_d,c)$

$$m=D(k_d, E(k_e, m))$$

# Digital signatures

Can be used to verify the source of a message.



- To send a signed message Alice encrypts the message  $m$  using her own private key( $Kd_A$ ).
- Bob decrypts the message using Alice's public key( $Ke_A$ ) – the two keys are interchangeable for the RSA algorithm.
- If the decryption process is successful then the message could only have come from Alice, the owner of the private key  $Kd_A$ .

# Hash functions

**Can be used to ensure message integrity.**

**Are also called One Way Functions or Message Digests.**

- Can be used to compute a signature (digest)  $MD(m)$  of a message  $m$  of any given length. The length of the digest is fixed.
- Examples of algorithms: MD5 (120b), SHA-1 (160b), SHA-256 (224, 256 or 512 bits)

# Digital certificates

A certificate acts as a trusted “third party” which allows the authentication of an entity even when it is the first contact with that entity.

- A certificate is issued by a Certificate Authority(CA)
- Digital certificates are created according to the X.509 standard.

I hereby certify that the public key

19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A  
belongs to

Robert John Smith

12345 University Avenue

Berkeley, CA 94702

Birthday: July 4, 1958

Email: bob@superduper.net.com

SHA-1 hash of the above certificate signed with the CA's private key

# Digital certificates

A certificate is not secret. It is signed by the CA which issued it.

Checking a certificate:

- The SHA-1 hash of the certificate is computed
- The signature of the certificate is decrypted using the public key of the CA.
- The two results are compared.

I hereby certify that the public key

19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A

belongs to

Robert John Smith

12345 University Avenue

Berkeley, CA 94702

Birthday: July 4, 1958

Email: bob@superduper.net.com

SHA-1 hash of the above certificate signed with the CA's private key

# Access control

# Methods of authentication

## Definition:

- Authentication is a method to determine the identity of a user.

There are 3 categories of authentication:

- **Ownership**: authentication based on an element owned by the user who must be authenticated
  - Ex: ID card, security token, etc.
- **Knowledge**: based on something known only by the user
  - Ex: username/password, gestures, PIN, etc.
- **Inherence**: based on an intrinsic property of the user
  - Ex: biometric authentication

## Methods of authentication

In general authentication takes one of the following three forms:

- **Basic authentication** involving a server. The server maintains a user file of either passwords and user names or some other useful piece of authenticating information. This information is always examined before authorization is granted.
- **Challenge-response**, in which the server or any other authenticating system generates a challenge to the host requesting for authentication and expects a response.
- **Centralized authentication**, in which a central server authenticates users on the network and in addition also authorizes and audits them.

# Example: Passwords

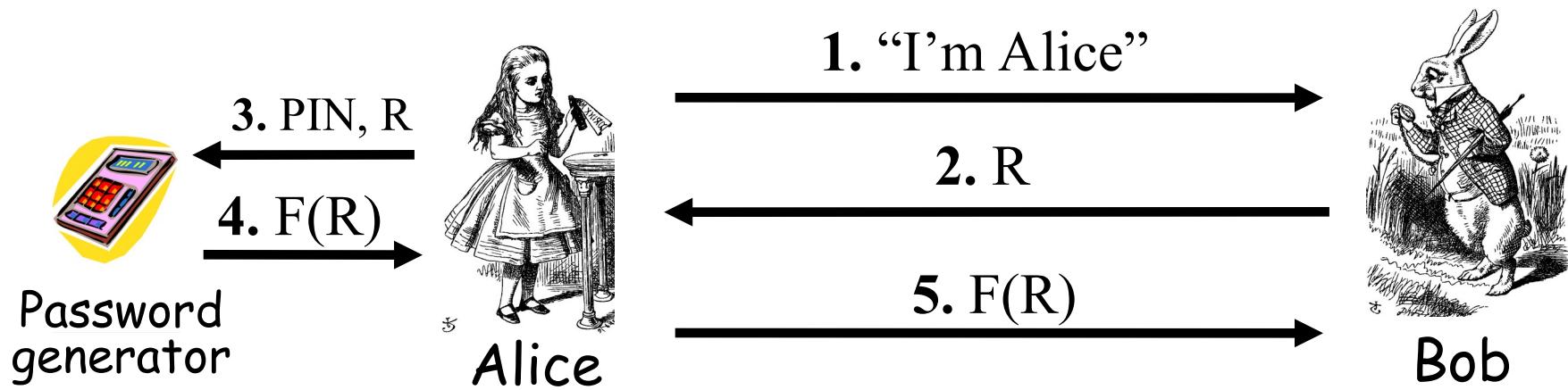
**The simplest method of authentication, based on a username and password**

- Simple to implement and verify
- Very low security
- Existing attacks:
  - Network traffic monitoring
  - Brute force
  - Dictionary attacks
  - Social engineering
- Possible improvements:
  - One Time Passwords
  - Physical token (password generator)

## 2-factor Authentication

- Requires 2 out of 3 of
  - Something you know
  - Something you have
  - Something you are
- Examples
  - ATM: Card and PIN
  - Credit card: Card and signature
  - Password generator: Device and PIN
  - Smartcard with password/PIN

# Password Generator



Alice gets “challenge” R from Bob

Alice enters R into password generator

Alice sends “response” back to Bob

Alice **has** a password generator and **knows** PINs

# Authorization

## Definition:

- determining a set of characteristics or access rights regarding certain resources for an entity whose identity has already been proven.

**Authentication + Authorization = Access Control**

## Minimum privilege principle:

- Only the minimum set of necessary access rights must be given to a user, in order of him to fulfill his function.

## Types of access control:

- **Discretionary Access Control (DAC)**
- **Mandatory Access Control (MAC)**
- **Non-Discretionary (RBAC)**

# Authorization

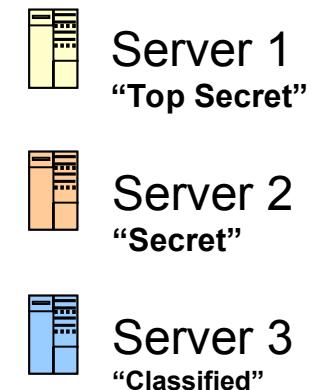
## • Discretionary Access Control (DAC)

- Allows the owner of a resource to specify the set of users who have access to that resource (access control is at the discretion of the owner).

Application Access List	
Name	Access
Tom	Yes
John	No
Cindy	Yes

## • Mandatory Access Control (MAC)

- Access control is based on a system of tags
- Each resource has an associated security level
- Each user has a certain access level



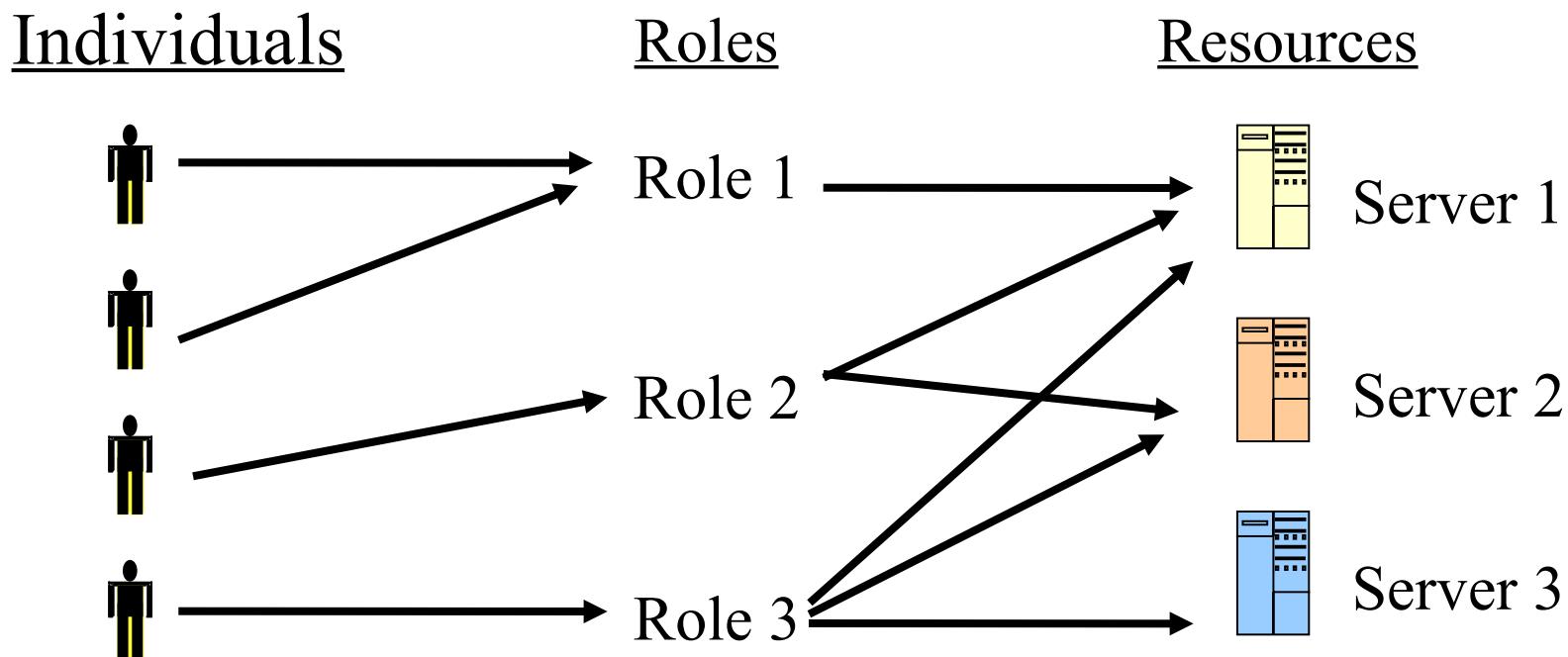
# An access matrix

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

# Role Based Access Control

- Each user has access to a resource based on a role which was assigned to him
- Roles are defined and assigned by a central authority
- Each role has a certain number of permissions



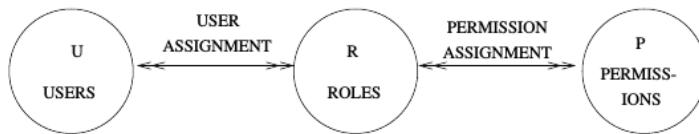
- Users may change frequently, roles usually don't

# Privilege

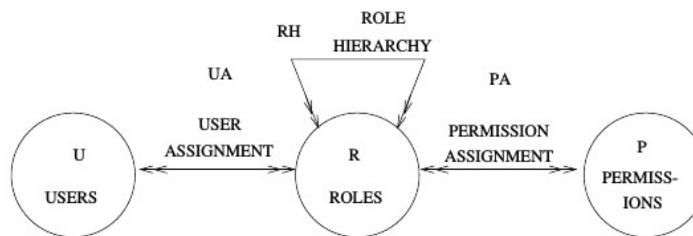
- Roles are engineered based on the **Principle of Least Privilege**.
  - A role contains the minimum amount of permissions to instantiate an object.
  - A user is assigned to a role that allows him or her to perform only what's required for that role.
  - No single role is given more permission than the same role for another user.

# Role Based Access Control

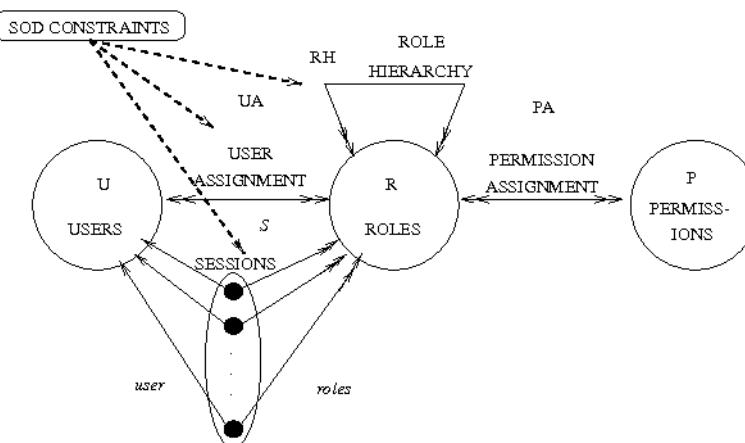
- Types of Role Based Access Control (RBAC)
  - Flat RBAC



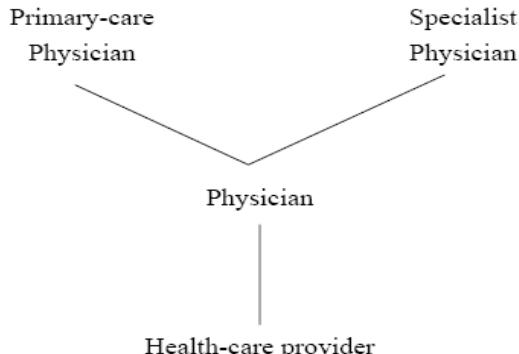
- Hierarchical RBAC



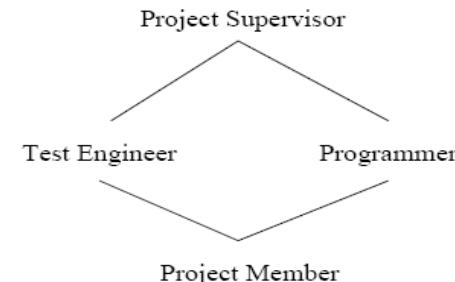
- RBAC with constraints



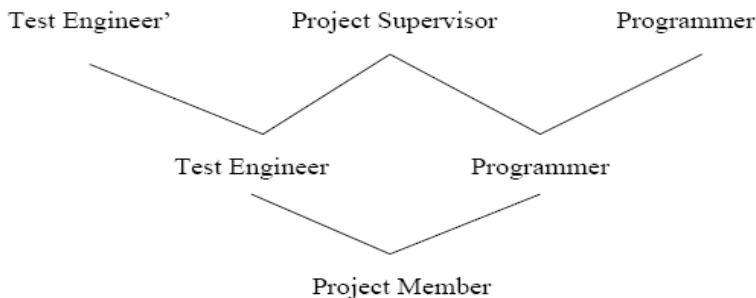
# Role Hierarchy Examples



(a)



(b)



# Constraints

- Is a convenience when RBAC is centralized
- When decentralized becomes a mechanism for restriction
- Types of Constraints
  - » Mutually exclusive roles/ permissions
  - » Cardinality constraints
  - » Prerequisite roles
- Effective only if suitable discipline is observed
  - » Mapping one user to more than one u-id
  - » Mapping one permission to more than one p-id
- Role Hierarchies can be considered a constraint

# The End

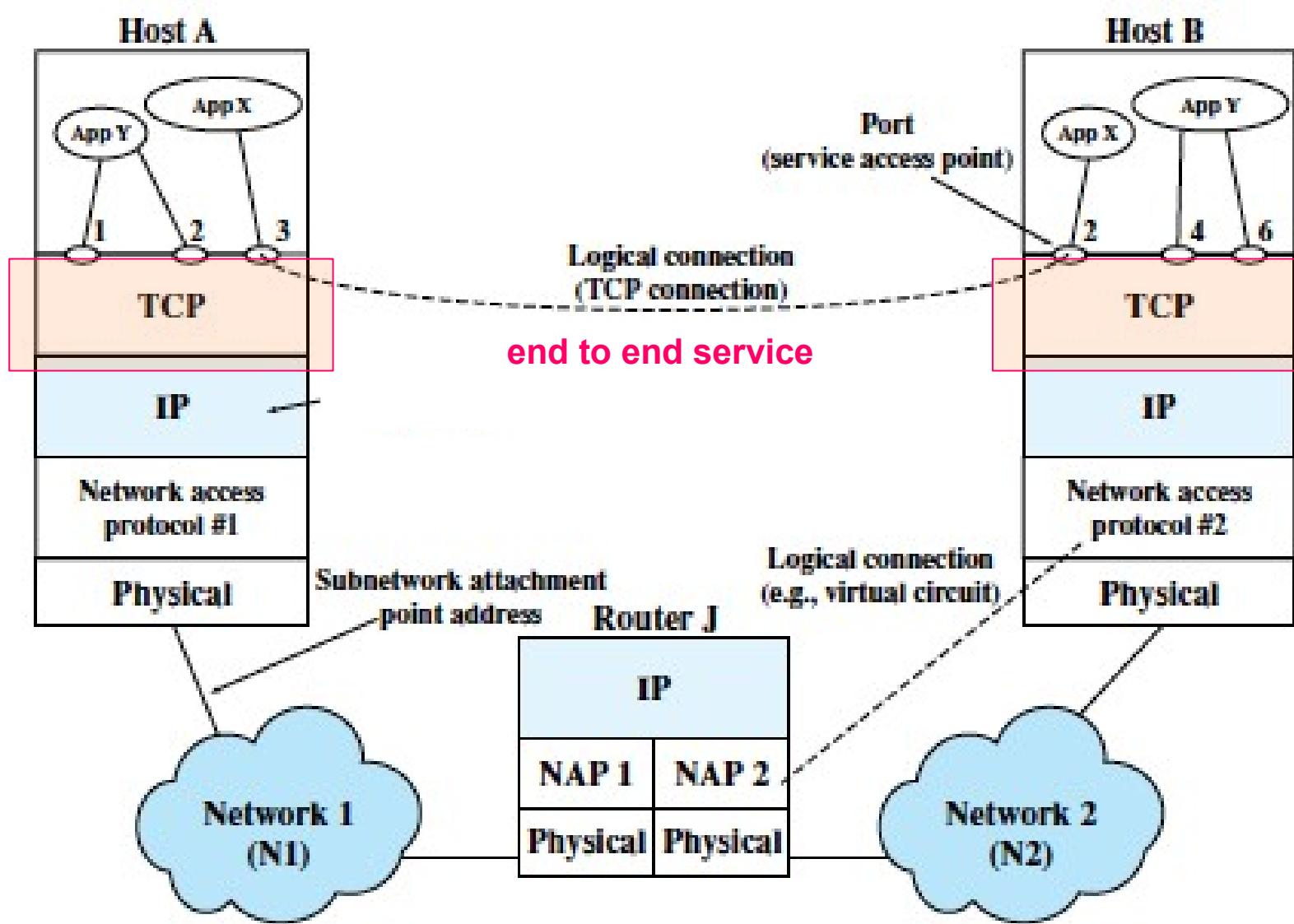
# Security Protocols

Lecture  
2

# Communication protocols

## Basic information

# Transport layer in TCP/IP

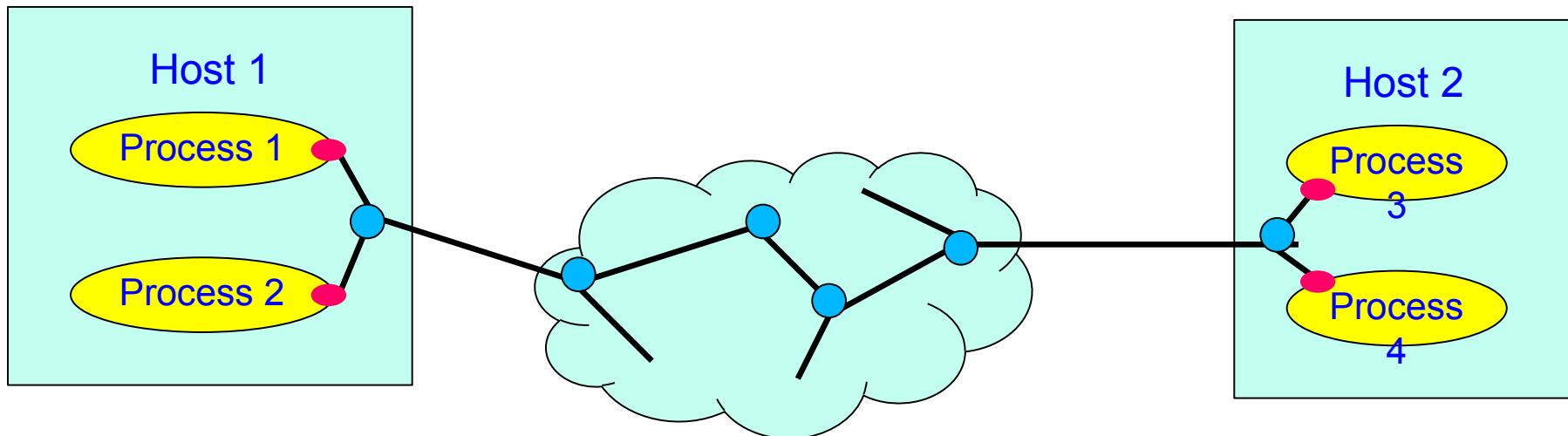


# Communication between applications

The network protocols ensure that packets are sent between different hosts.

The transport layer ensures communication between processes

- Creates a link between processes and network access points
- Identifying an access point through **<network address, port>**



# Transport level Services

- **Offered services**
  - Data transfer between processes, through different types of network
  - Common user interface
- **Characteristics**
  - Two types of services:
    - connection oriented - TCP
    - connectionless - UDP

# Socket interface

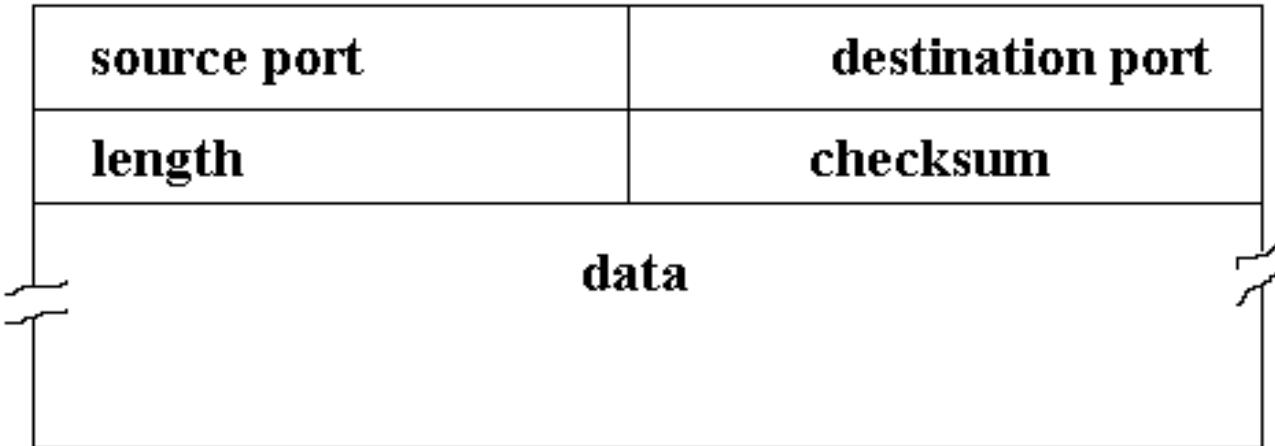
- Transport level services are offered as an API
- *Socket API*
  - Originally part of Berkeley BSD UNIX
  - Available also on Windows, Linux, Solaris, etc.
  - **socket** = the endpoint of an application,  
which connects to the network
  - Identified by a file descriptor
- It is a *de facto* standard



# UDP - User Datagram Protocol

- UDP delivers *user datagrams*
  - “Best effort” delivery – datagrams can be lost, received in a different order, corrupted etc.
  - Uses checksums of integrity verification
- UDP endpoints = *protocol ports* or *ports*
- UDP identifies the endpoints using the *internet address* and port number
- *Destination port* and *source port* can be different.

# UDP Header



Checksum used in the same way as TCP

Has no flow control

Has no error control

Doesn't support frame retransmission

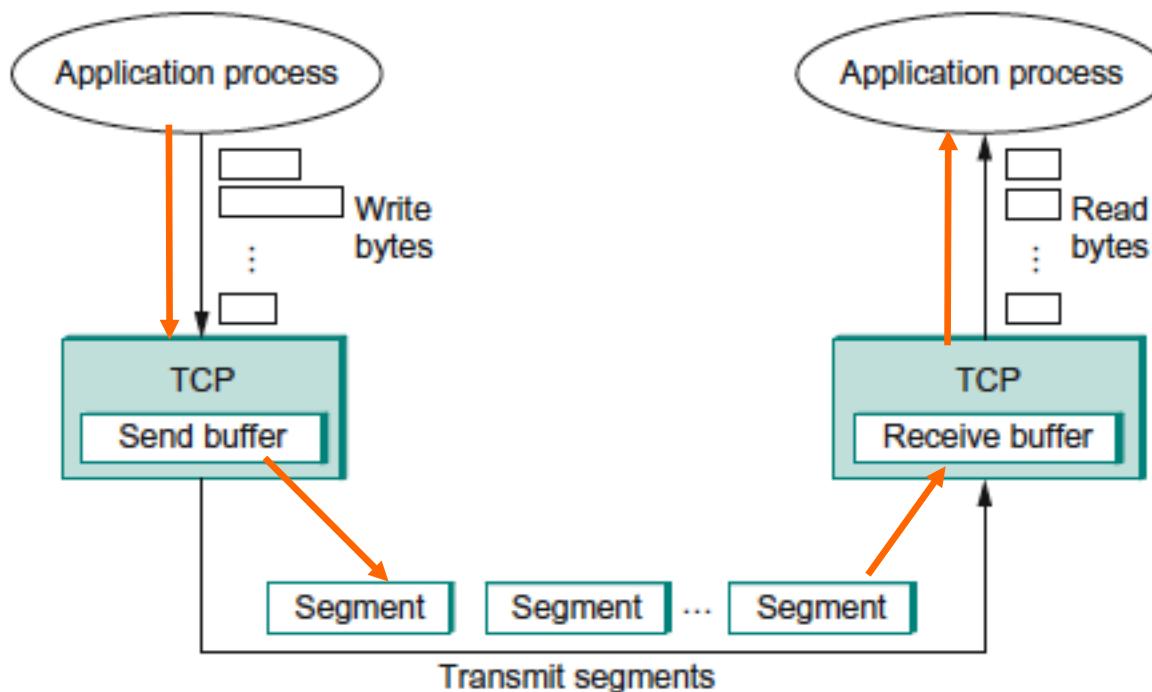
# Applications using UDP

- DNS
- Voice over IP
- Online Games
- Can be used when:
  - Latency is very important
  - Delivering all the packets is not essential
  - Resending corrupted packets is implemented by the application (DNS)

# Standard ports

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

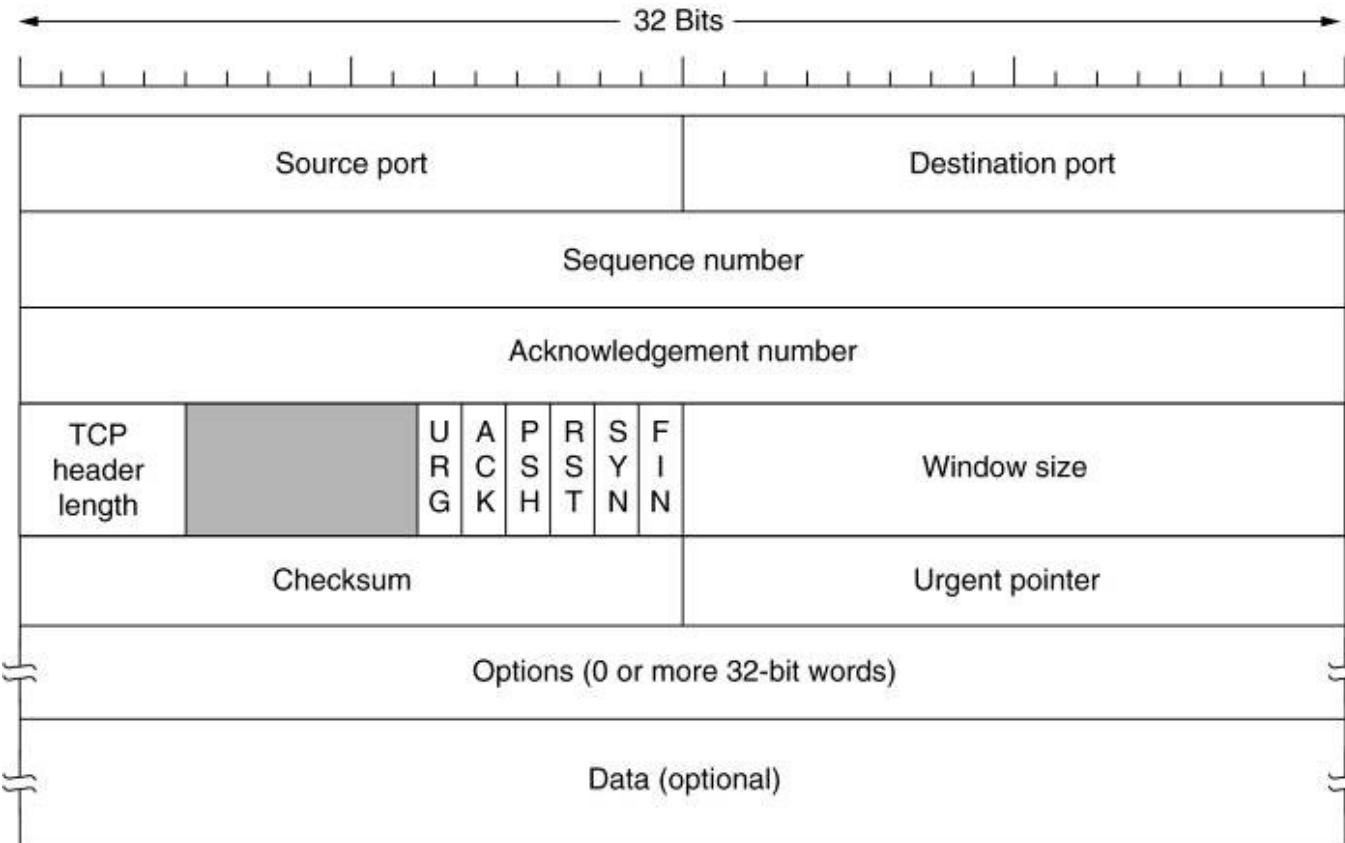
# TCP is a connection oriented protocol



# Characteristics

- Connection oriented
- Handles the reception and sending of byte flows
- Handles congestion control by adapting the transmission rate to network conditions
- Guarantees that data sent is received in the same order as part of a connection
- **Full duplex**
- Secure connection creation - three-way handshake
- Secure connection release – without data loss

# Header for a TCP segment



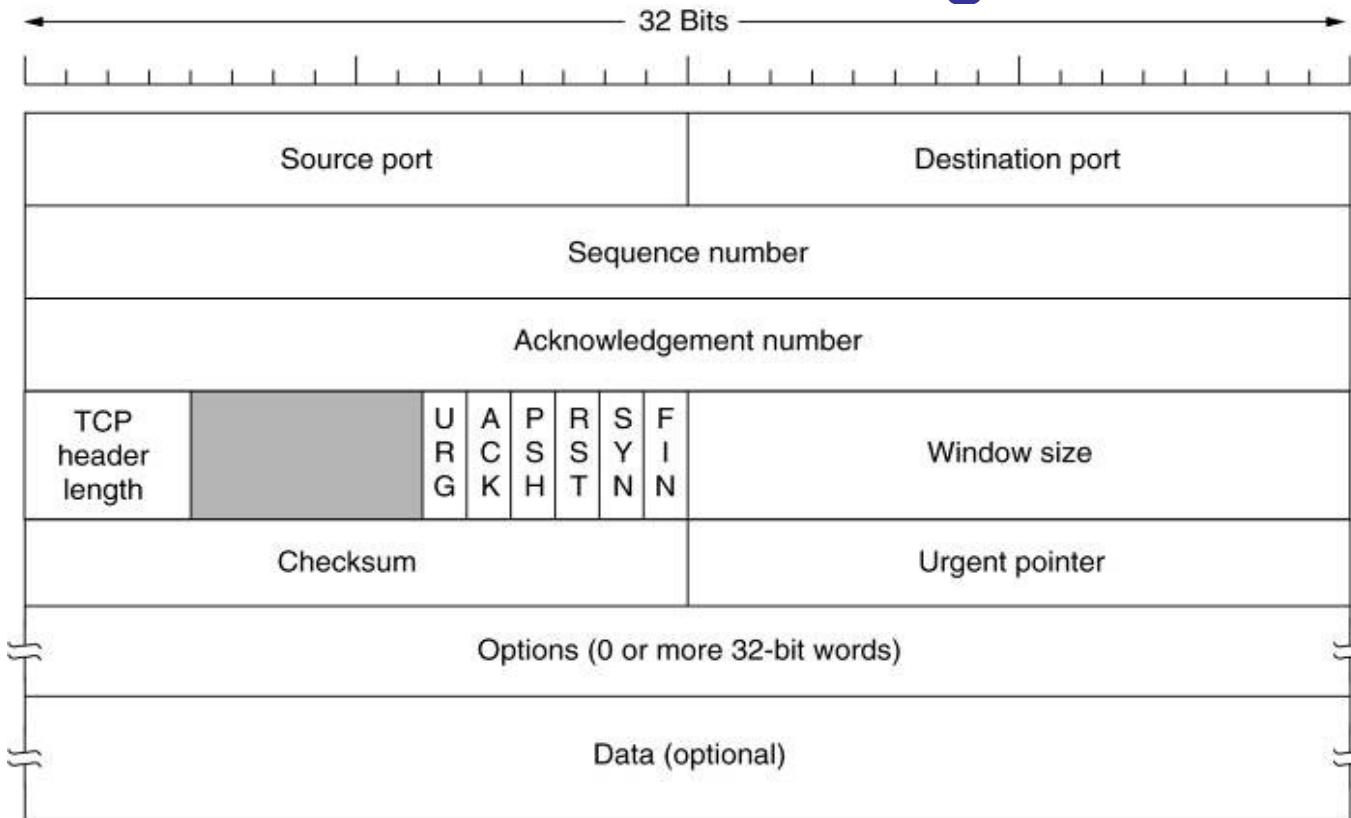
**Sequence number** – the id number of the first byte in the segment

**Acknowledgement number** – The id of the next expected byte

**Window size** – The number of bytes that can be sent

**Urgent Pointer** – *offset*, as opposed to the sequence number

# Header for a TCP segment



**URG** Urgent pointer valid

**ACK** Acknowledge Number valid

**PSH** - push information to user

**RST** - close a connection due to an error

**SYN** - open connection

**FIN** - close a connection

**Options**: e.g. max TCP payload (default **536 bytes**), selective repeat

# PPP, PAP, CHAP

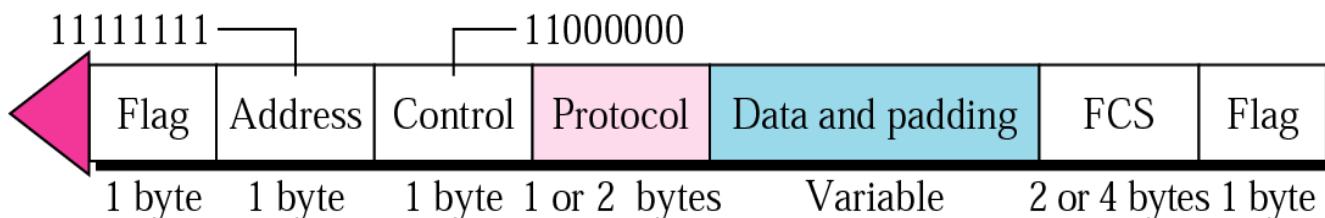
# Point-to-Point Protocol (PPP)

- Two devices can be connected by a dedicated link or a shared link.
- Dedicated link between two devices → point-to-point access.
- Connections to ISP via traditional modem or cable modem or DSL modem requires some protocol to manage and control the transfer the data. These modems provide physical connections only.
- PPP Services
  - Format of the frame
  - Negotiate the establishment of the link and the exchange of data
  - How network layer data are encapsulated in the data link frame
  - How two devices can authenticate each other.

# PPP Frame

## Frame Format:

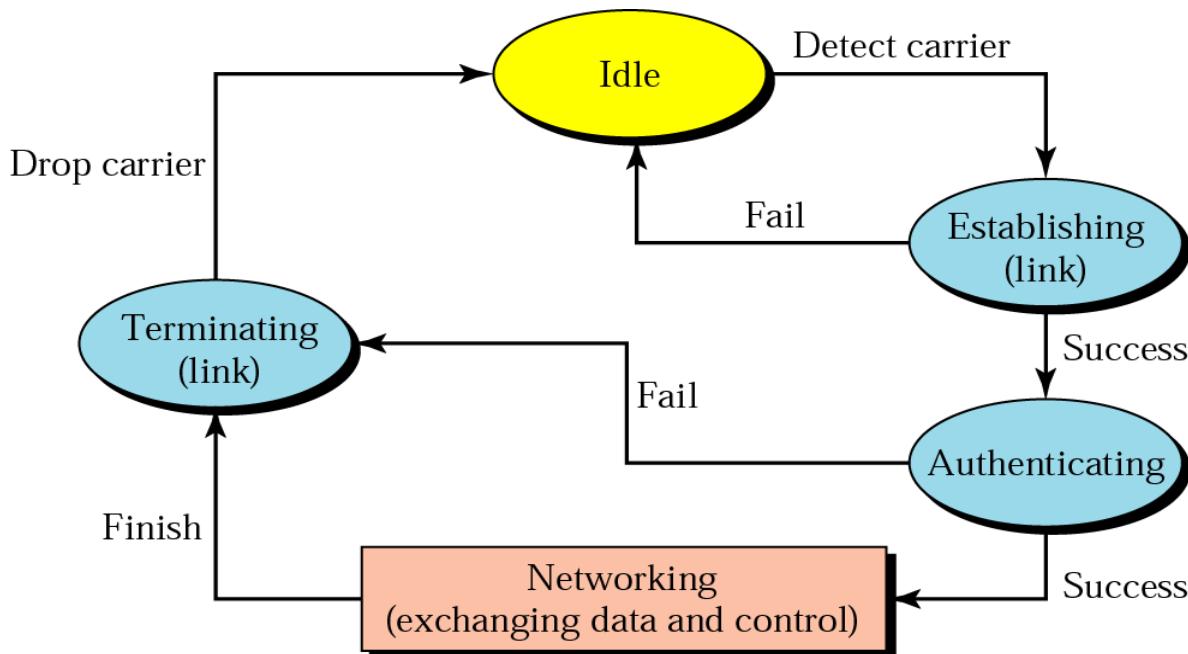
- Flag = 01111110; Identify the boundary of PPP frame.
- Address is always Broadcast because of point-to-point link.
- Control field [11000000] = No sequence numbers and so no flow or error control
- Protocol field = What is being carried in the data field: user data or other information.
- FCS [Frame Check Sequence] = 2-byte or 4-byte CRC



# Transition states

Idle: No active carrier, line is quiet.

Establishing: When one endpoint starts communication, connection goes to establishing state. Negotiation of options.

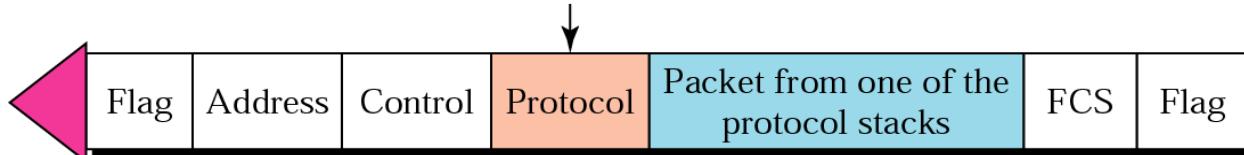


# Protocol stack

- Link Control Protocol (LCP)

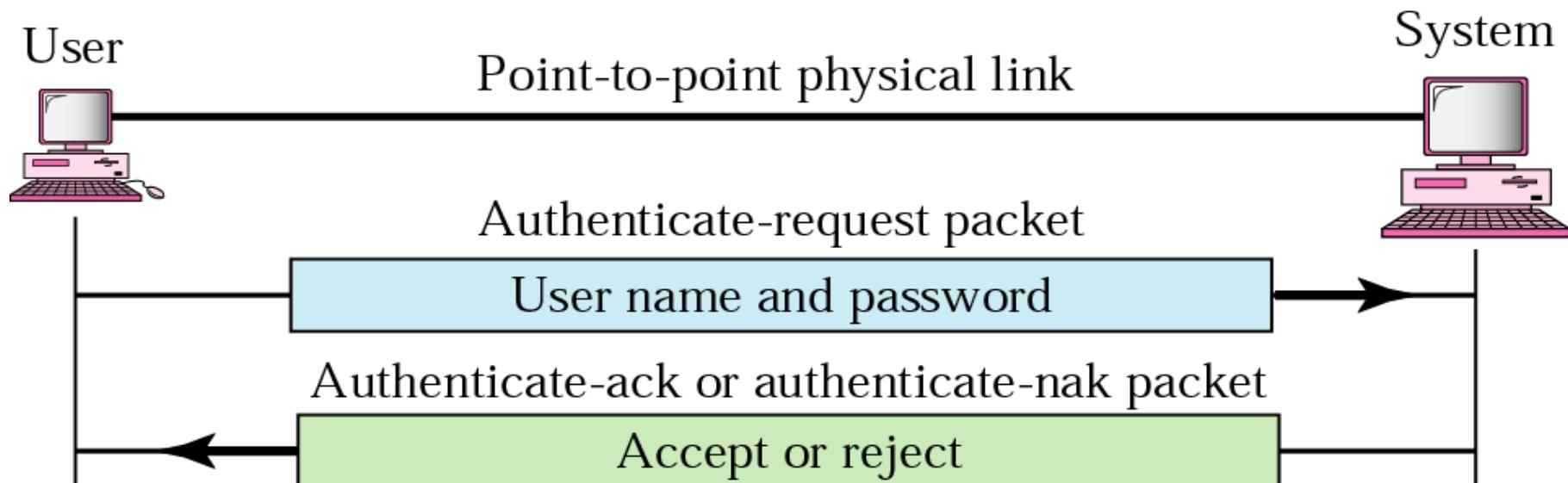
- Responsible for establishing, maintaining, configuring, and terminating links.
- Negotiation mechanisms.
- LCP occurs in Establishing or terminating state
- LCP packets are carried in the data field of PPP frames.
- Authentication Protocols
- Password Authentication Protocol (PAP)
- Challenge Handshake Authentication Protocol (CHAP)
- Network Control Protocol

The value of the protocol field defines the protocol stack.



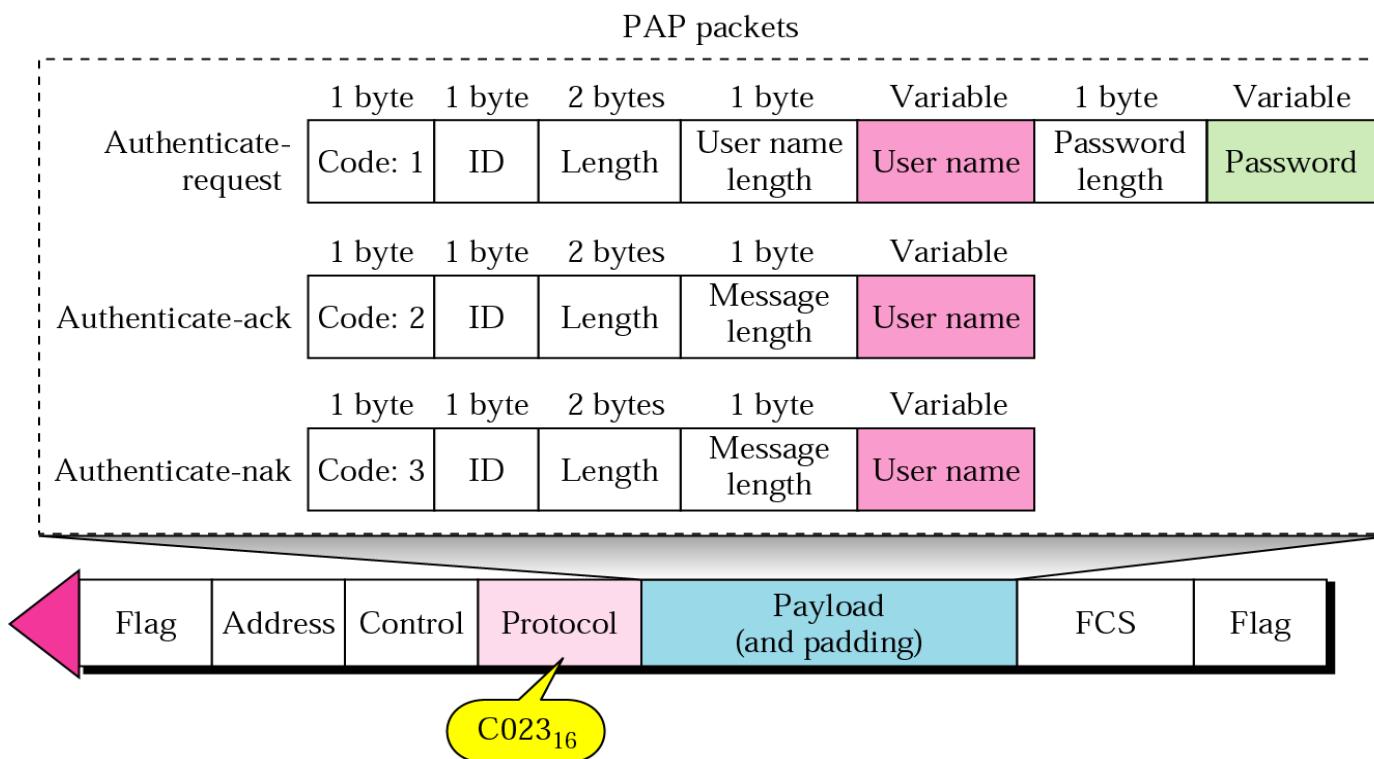
# Password Authentication Protocol (PAP)

- Send user name and password
- Check validity and accept or deny.
- Access to the link can get the user name and password.



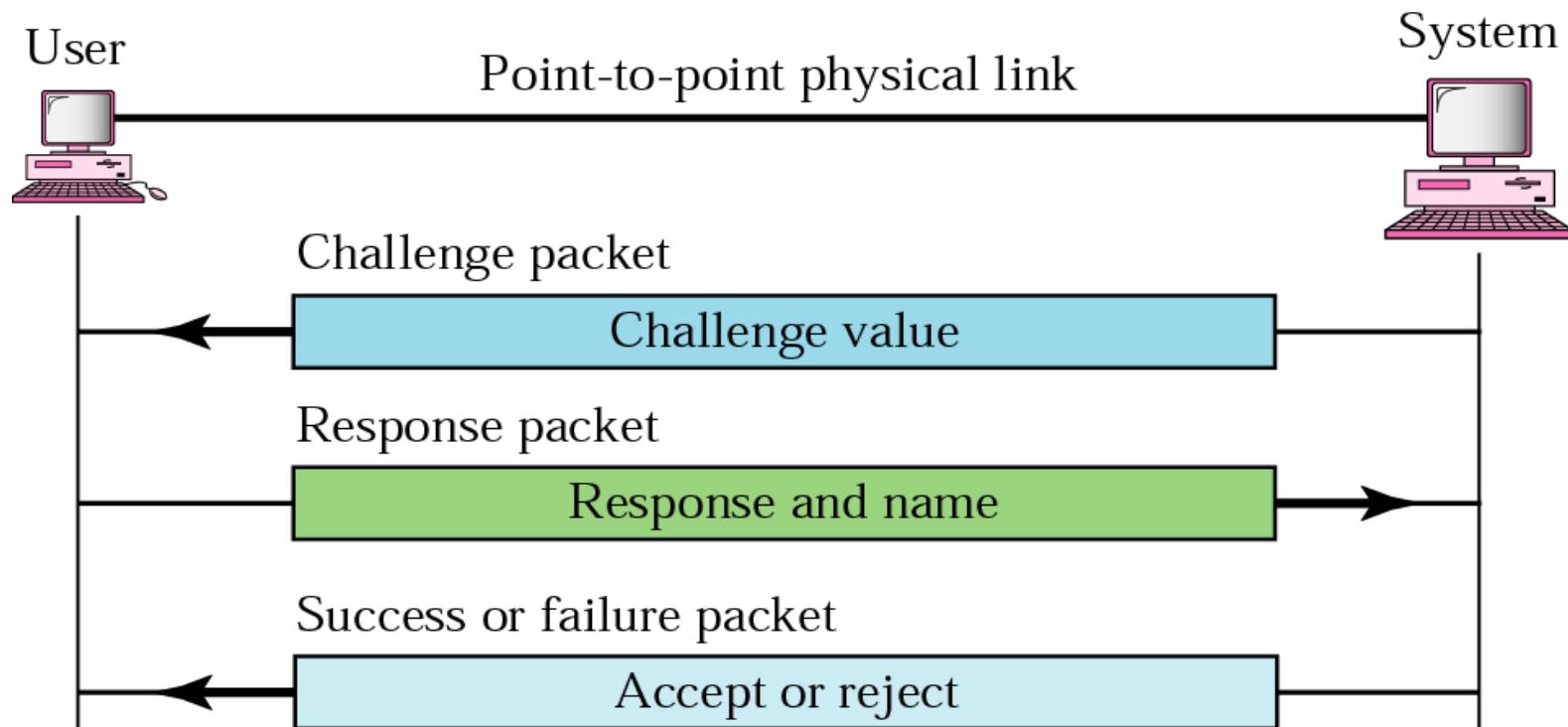
# PAP packets

- PAP has protocol field value of C023 in PPP frame.
- Three PAP-packets: authenticate-request, authenticate-ack and authenticate-nak.



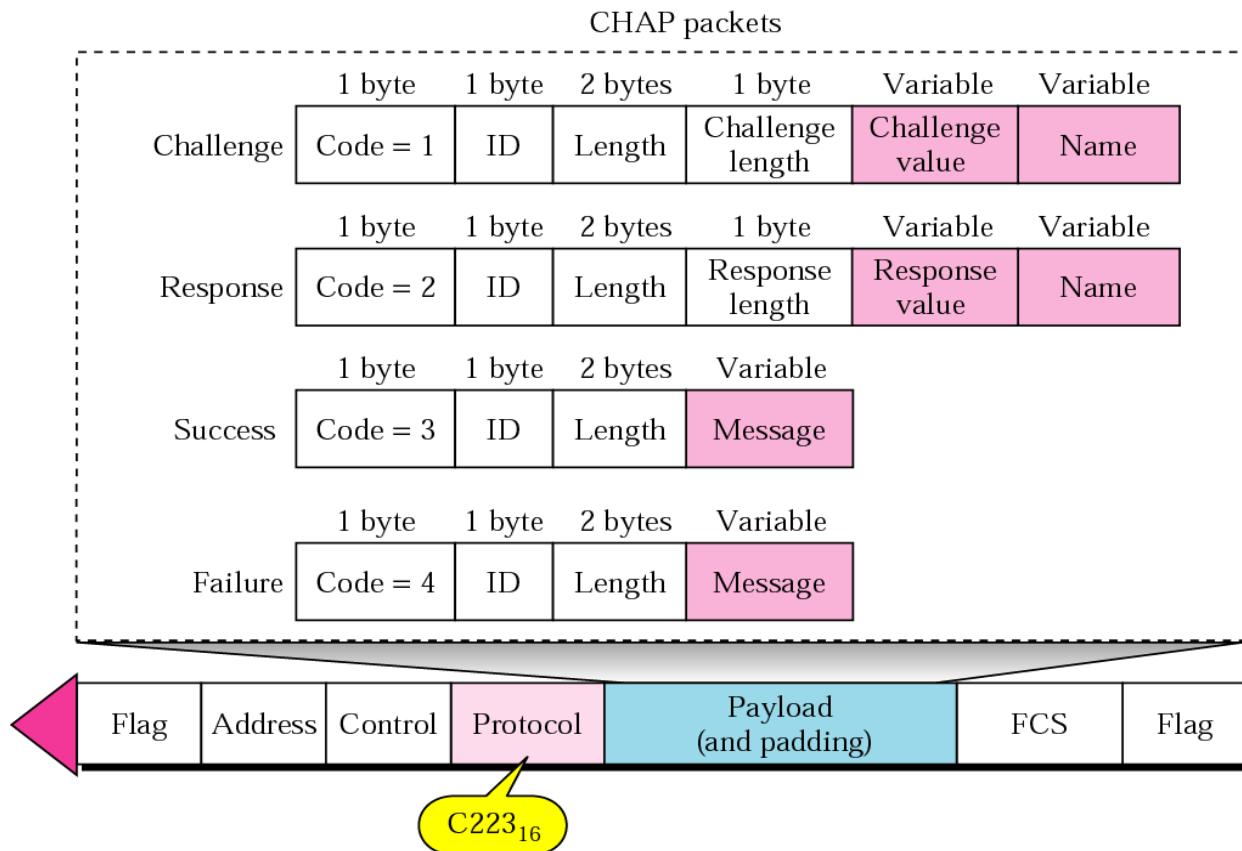
## Challenge Handshake Authentication Protocol (CHAP)

- Three-way handshake. Password is never sent online.
- System sends a challenge packet containing challenge value, usually a few bytes
- User applies predefined function with challenge value, user's password. The result is sent as response.
- System does the same as user and compares the result.

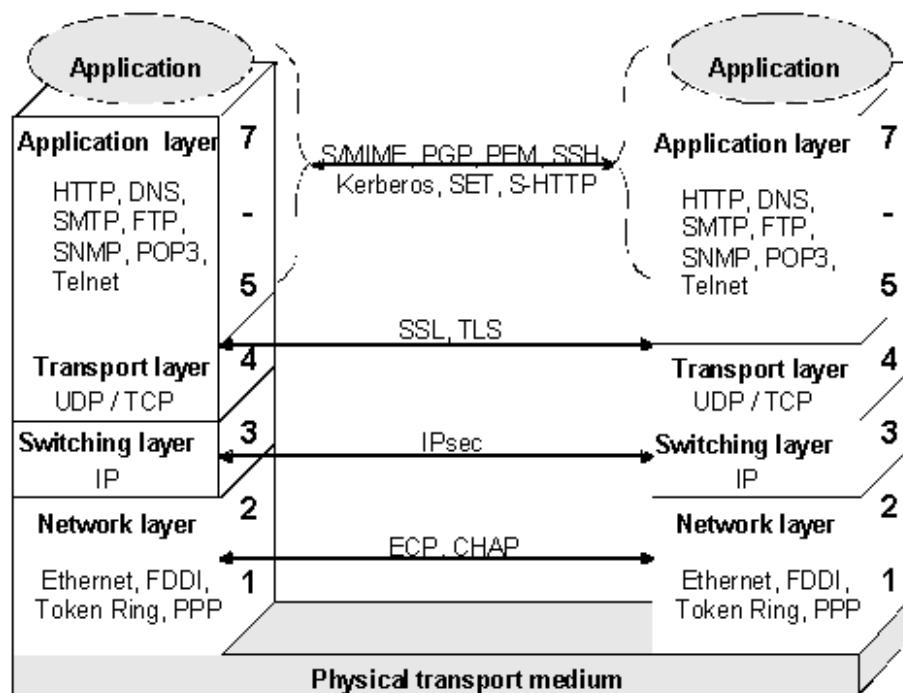


## CHAP packets

- CHAP has protocol field value of C223 in PPP frame.
- Four CHAP-packets: challenge, response, success, failure.



## Overview of security mechanisms



Cryptographic procedures are used for securing a variety of information that arises in the course of communication, i.e. for encrypting information or for assigning cryptographic checksums or digital signatures to it.

The data that can be secured is the data to be transmitted by the user, but also information that is generated implicitly during the exchange of information (such as traffic flow information).

Source: <http://www.iwar.org.uk/comsec/resources/standards/germany/itbpm/s/s4090.htm>

## Overview of security mechanisms

Use of cryptographic procedures on	
higher layers:	lower layers:
<ul style="list-style-type: none"> <li>+ Makes sense if the application data is to be protected close to the application or if the "insecure channel" is to be kept as short as possible</li> <li>+ Whenever the data is not protected on the lower layers</li> <li>+ Makes sense if there are a large number of changing communication partners at various locations</li> <li>+ Users can make use of them in accordance with their own requirements</li> <li>+ Security is provided closer to the user, and is more easily recognisable by him</li> <li>- Undermines safeguarding by firewalls</li> <li>- Often subject to operating errors</li> <li>- Often based on software; cryptographic keys and algorithms are easier to manipulate</li> <li>- Greater dependency on the operating system or on underlying hardware</li> </ul>	<ul style="list-style-type: none"> <li>+ Makes sense for the coupling of two networks that are classed as secure via an insecure connection, e.g. linking two properties via public networks</li> <li>+ For securing a network against unauthorised access</li> <li>+ Whenever traffic flow information needs to be protected, e.g. address information</li> <li>+ All higher-level header information and the user information is encrypted</li> <li>+ Transparent for users, relatively low risk of operating error</li> <li>+ Easier key management</li> <li>- Protection only as far as the layer in which the security protocols are implemented</li> <li>- Often hardware, i.e. expensive and inflexible</li> <li>- Often does not offer end-to-end security</li> </ul>

Source: <http://www.iwar.org.uk/comsec/resources/standards/germany/itbpm/s/s4090.htm>

# IP security



# IP Security Protocol - IPsec

- Implemented at the IP level
- Creates a secure unidirectional link between the sender and the receiver
  - Named **Security Association** - SA
  - Ensures one of the following:
    - Message authentication
    - Authentication and encryption
- Two-way secure communication => 2 x SA



## Security parameters

- SA is not tied to a single encryption algorithm. The protocol can specify:
  - The encryption algorithm(ex. DES in block-chaining mode)
  - Encryption key
  - Encryption parameters (ex. Initialization Vector)
  - Authentication protocol
  - Time-to-live for a SA (allows long sessions by regularly changing the key if necessary)
  - Address for the other end of a SA
  - Sensitivity level of protected data



# SA Database

A system creates a database for all security associations in use

- Security parameters on the previous slide, plus:
- **Counter** for sequence numbers
- **overflow** indicator for sequence number counters: what can be done if the counter limit is exceeded
- **anti-replay** window: determines if a packet is a copy of a previous one
- **Path MTU**: path Maximum Transmission Unit (to avoid fragmentation)



## SA Database (2)

Each entry is uniquely identified using:

- **Security Parameters Index (SPI)**
- **IP Destination Address**
- **Security Protocol Identifier**

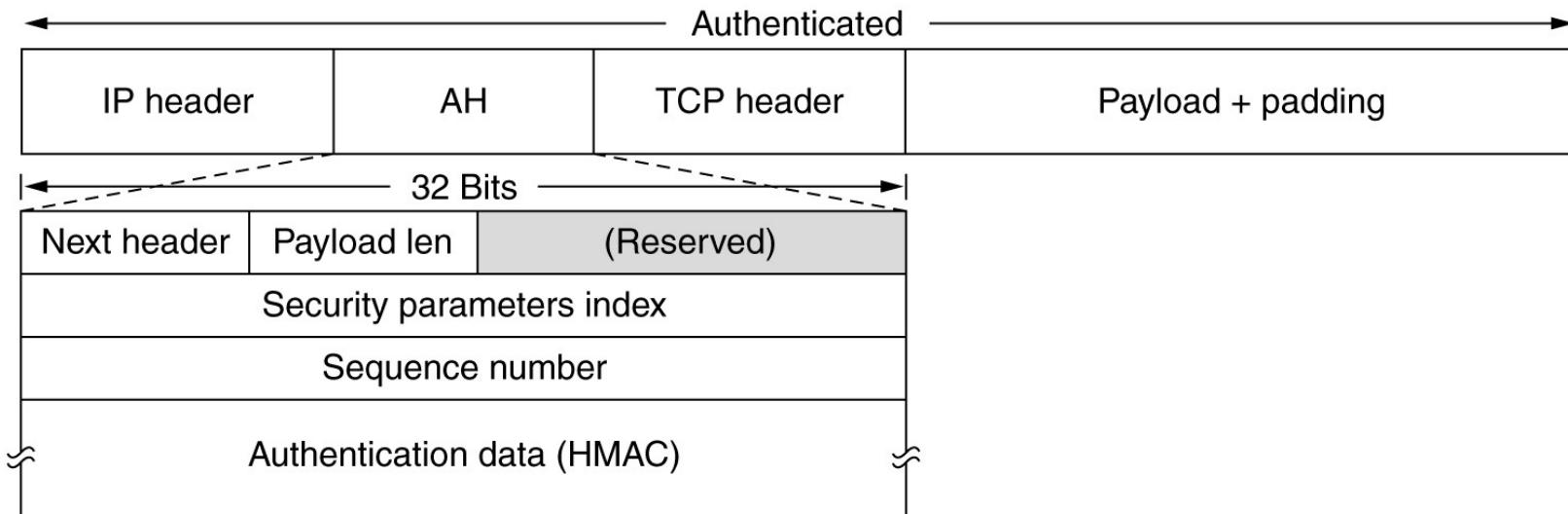
Two security protocols:

- AH ([Authentication Header](#)) – authentication protocol
- ESP ([Encapsulating Security Payload](#)) – authentication + encryption

Two operation modes

- transport
- tunnel

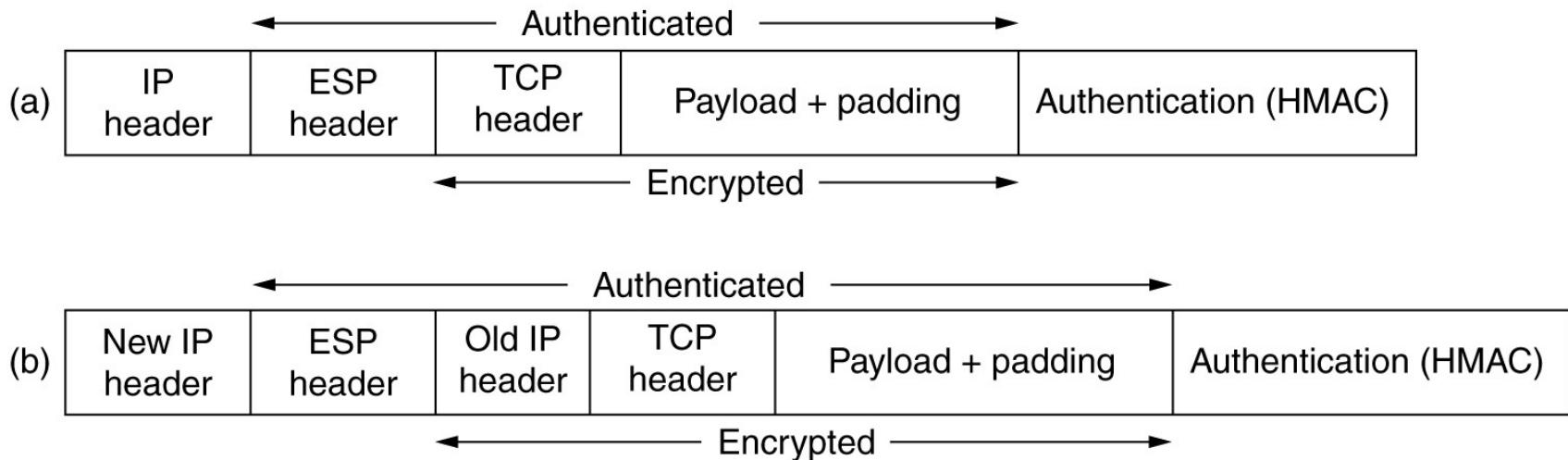
# AH protocol – in transport mode for IPv4



**Authentication Header** – inserted into an IP datagram

- **Next header** – taken prof **IP header**, where it is replaced by 51
- **Payload len** – AH length (number of 32 bit words) minus 2
- **Security Parameters Index** – the index in the receivers SA database
- **Sequence number** – to avoid replica attacks
- **HMAC** – Hashed Message Authentication Code
  - Uses a symmetric key
  - Computes the hash for the entire datagram + symmetric key

# ESP in transport and tunnel modes



## ESP – Encapsulating Security Payload

### (a) ESP in transport mode.

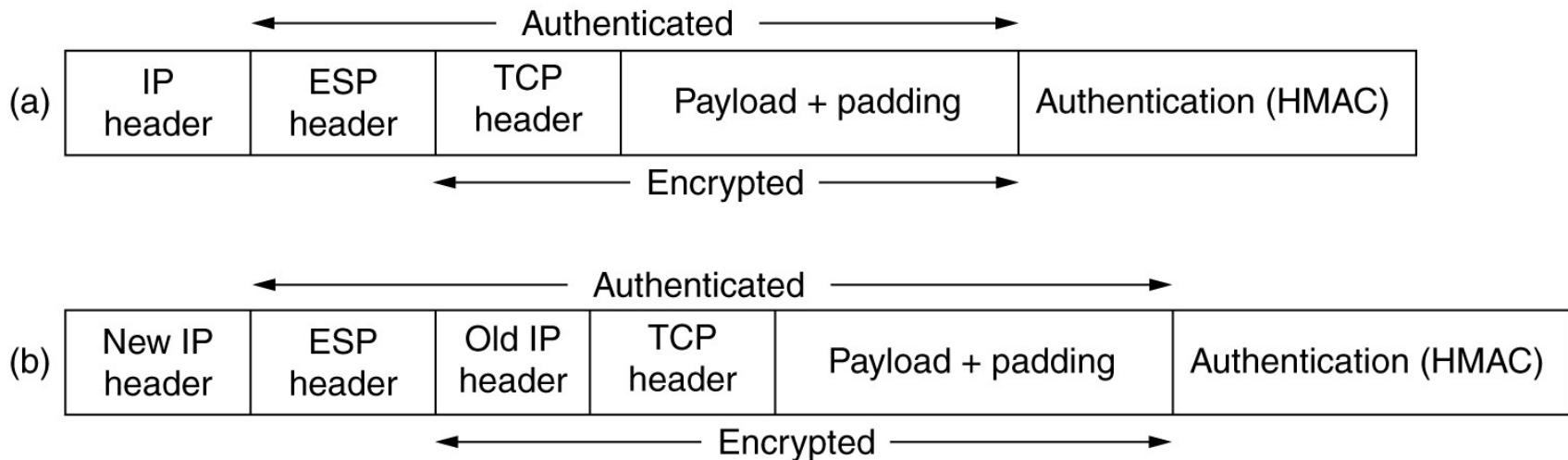
- ESP header is placed between the IP and TCP headers
- The “protocol” field from the IP header is modified to show that it is followed by an IPsec header

### (b) ESP in tunnel mode.

- The IPsec header is added with a new IP header
- The tunnel can end before the final destination (ex. a firewall)



# ESP in transport and tunnel modes



# ESP – Encapsulating Security Payload

(a) ESP transport mode. (b) in tunnel mode.

- The payload is protected by encryption;
  - Authentication protects the ESP header + payload

# ESP header includes

# Security Parameters Index

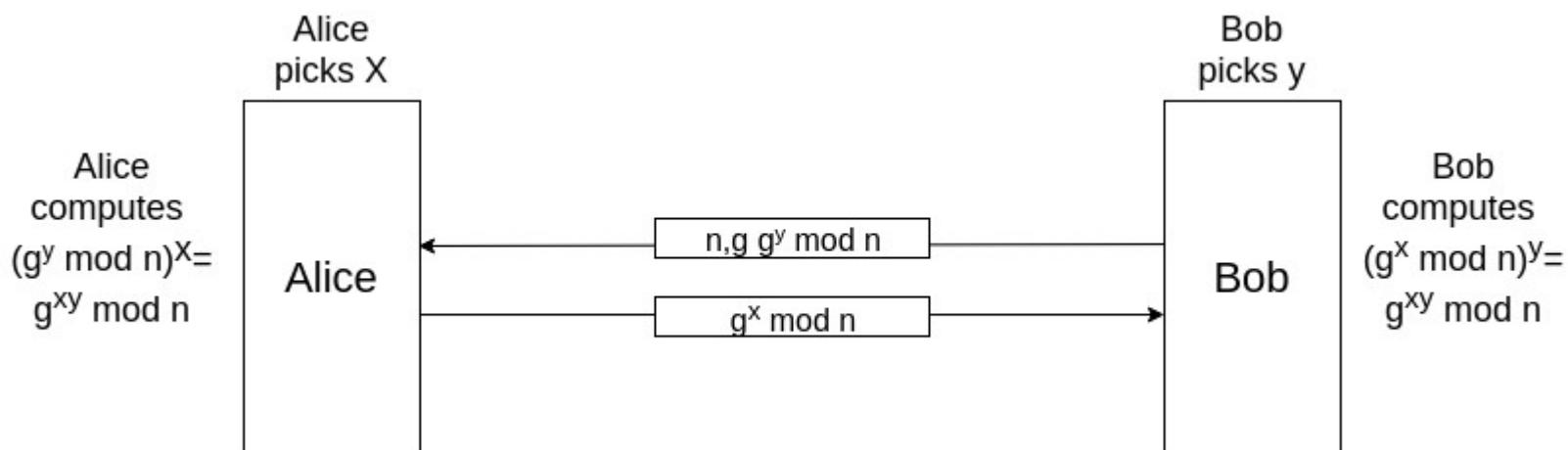
## Sequence number

Initialization vector (for data encryption)

# HMAC – Hashed Message Authentication Code

# Key management

- ISAKMP – Internet Security Association Key Management Protocol
- Generates a distinct key for each SA
- Implemented using IKE (Internet Key Exchange)
  - Uses Diffie – Hellman
- Diffie-Hellman key exchange:
  - $x$  is the private key
  - $g^x \text{ mod } n$  is the public key
  - $K_{A,B} = g^{xy} \text{ mod } n$  is the secret key shared with Bob



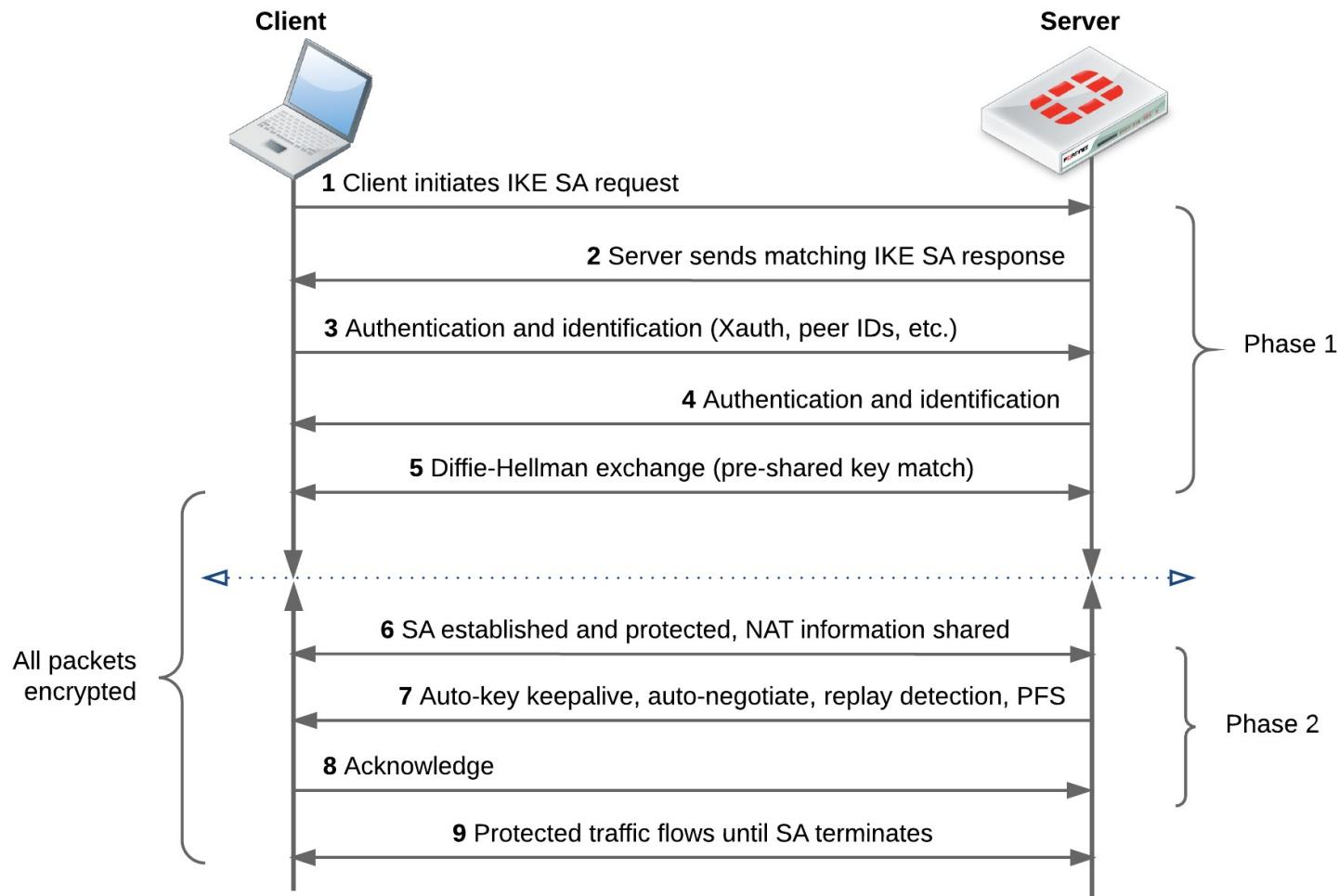


# IKE

- IKEv1
  - Phase 1:
    - A client initiates the IKE Security Association (SA) request
    - Identities are sent (IDs, certificates, etc.)
    - Contains the Diffie-Hellman Key Exchange
  - Phase 2:
    - The SA is established and protected
    - SA parameters are renegotiated after a set duration

# IKE

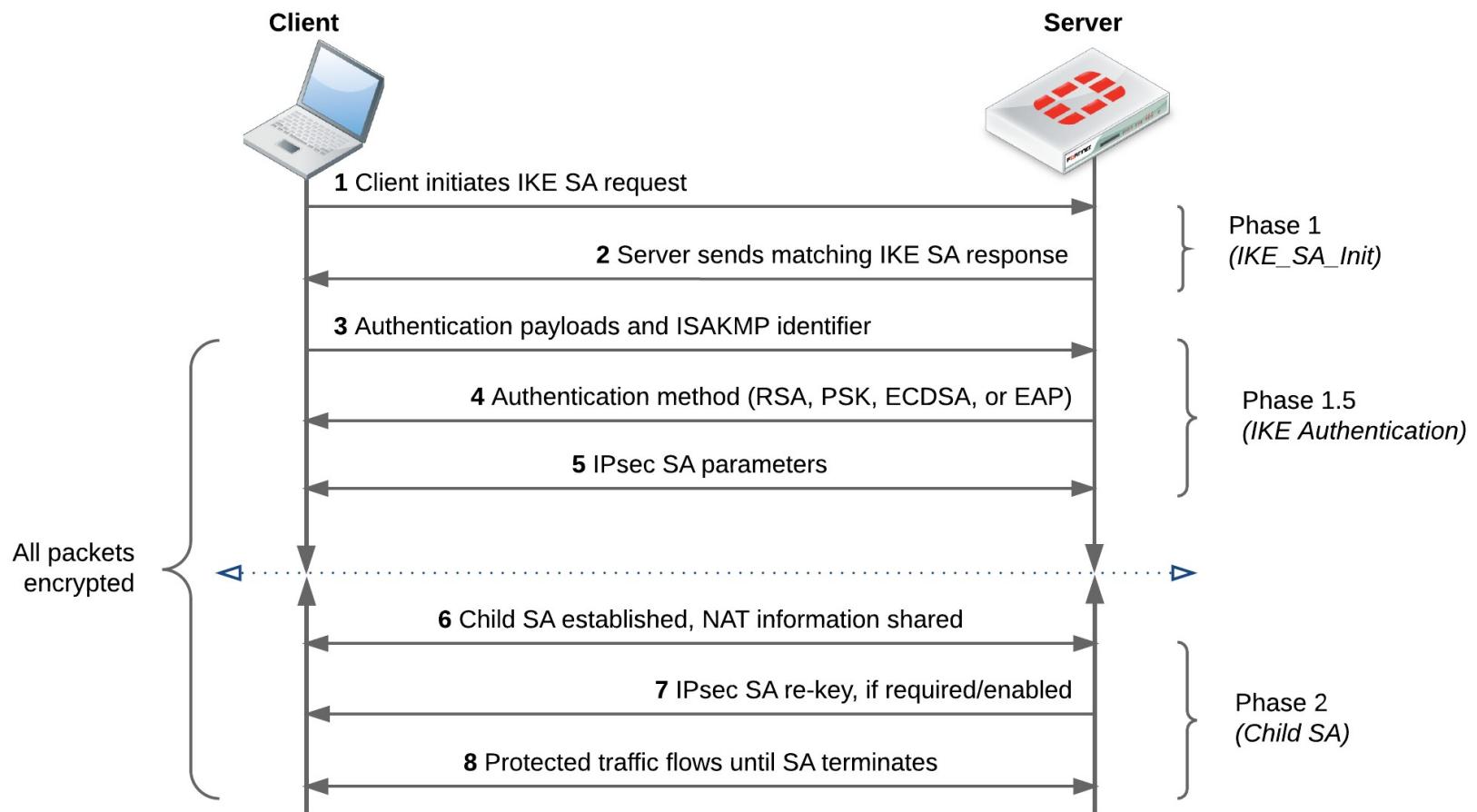
- IKEv1



# IKE

- IKEv2

- The authentication step is also encrypted
- Can support different authentication protocols
- Authentication is a separate step (1.5)





# Characteristics of IPsec

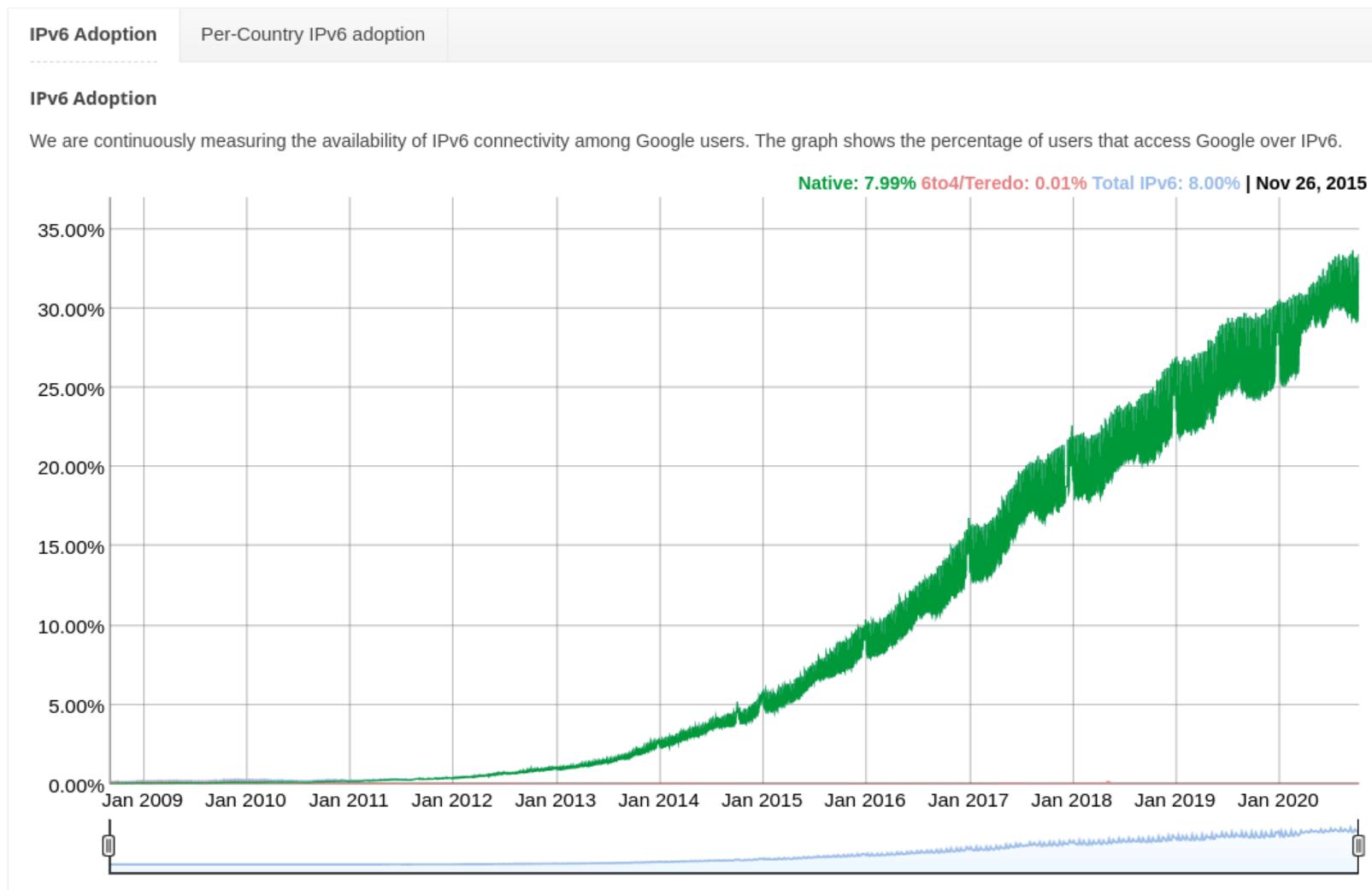
- IPSec is connection oriented
- Allows the selection of different algorithms for:
  - encryption: DES in mode CBC, 3DES, IDEA, ...
  - Authentication: MD5, SHA (trunchiat la 96 biti)
  - New algorithms can be added
- Allows encryption key exchange
- Allows the selection of different security services
  - Confidentiality
  - integrity
  - Replica attack protection

# IPv6

## What is “IPv6” about?

- IPv6 was developed to address the exhaustion of IPv4 addresses
- Addresses have 128b length
- IPv6 has not yet seen broad/global deployment (current estimations are that IPv6 traffic is less than 30% of total traffic)
- However, general-purpose OSes have shipped with IPv6 support for a long time – hence part of your network is already running IPv6!
- Additionaly, ISPs and other organizations have started to take IPv6 more seriosly.

# Ipv6 adoption rate



Source: <https://www.google.com/intl/en/ipv6/statistics.html>

# Some interesting aspects of IPv6 security

- There is much less experience with IPv6 than with IPv4
- IPv6 implementations are less mature than their IPv4 counterparts
- Security products (firewalls, NIDS, etc.) have less support for IPv6 than for IPv4
- The complexity of the resulting network will increase during the transition/co-existance period:
  - Two internetworking protocols (IPv4 and IPv6)
  - Increased use of NATs
  - Increased use of tunnels
  - Use of other transition/co-existance technologies
  - Lack of well-trained human resources

# The IPv6 Header

40 Octets, 8 fields



# IPv6 Addressing

- IPv6 Addressing rules are covered by multiples RFC's
  - Architecture defined by RFC 2373
- Address Types are :
  - Unicast : One to One
  - Anycast : One to Nearest
  - Multicast : One to Many
  - Reserved
- A single interface may be assigned multiple IPv6 addresses of any type (unicast, anycast, multicast)
  - No Broadcast Address -> IPv6 Use Multicast

# Security Advantages of IPv6 Over IPv4

**IPv4 - NAT breaks end-to-end network security**

**IPv6 - Huge address range – No need of NAT**

**IPv4 – IPSEC is Optional**

**IPv6 - Mandatory in v6**

**IPv4 - Security extension headers(AH,ESP) – Back ported**

**IPv6 - Built-in Security extension headers**

**IPv4 - External Firewalls introduce performance bottlenecks**

**IPv6 - Confidentiality and data integrity without need for additional firewalls**

# Important Security fields in IPv6

- IPV4 - Security option field and Optional IPSEC
- IPV6 - IPSEC part of protocol suite-mandatory  
IPSEC provides network-level security
- IPSEC uses:-  
AH ( Authentication Header)  
ESP( Encapsulating Security Payload) Header

# Brief overview and considerations

- Myth: “*IPv6 is more secure than IPv4 because security was incorporated in the design of the protocol, rather than as an ‘add-on’*”
- This myth originated from the fact that IPsec support is mandatory for IPv6, but optional for IPv4
- In practice, this is irrelevant:
  - What is mandatory is IPsec support – not IPsec usage
  - And nevertheless, many IPv4 implementations support IPsec, while there exist IPv6 implementations that do not support IPsec
  - Virtually all the same IPsec deployment obstacles present in IPv4 are also present in IPv6
- The IETF has acknowledged this fact, and is currently changing IPsec support in IPv6 to “optional”
- Conclusion: there is no reason to expect increased use of IPsec as a result of IPv6 deployment

# Security considerations

- Transition technologies increase the complexity of the network, and thus the number of potential vulnerabilities.
- Many of these technologies introduce “Single Points of Failure” in the network.
- Transition/co-existance traffic usually results in complex traffic (with multiple encapsulations).
- This increases the difficulty of performing Deep Packet Inspection (DPI) and (e.g. prevent the enforcement of some filtering policies or detection by NIDS).

# The End

# Security Protocols

Lecture  
3

# Firewalls

# Firewalls

- Block/filter/modify traffic at network-level
  - Limit access to the network
  - Installed at perimeter of the network
- Why network-level?
  - Vulnerabilities on many hosts in network
  - Users don't keep systems up to date
  - Lots of patches to keep track of
  - Zero-day exploits

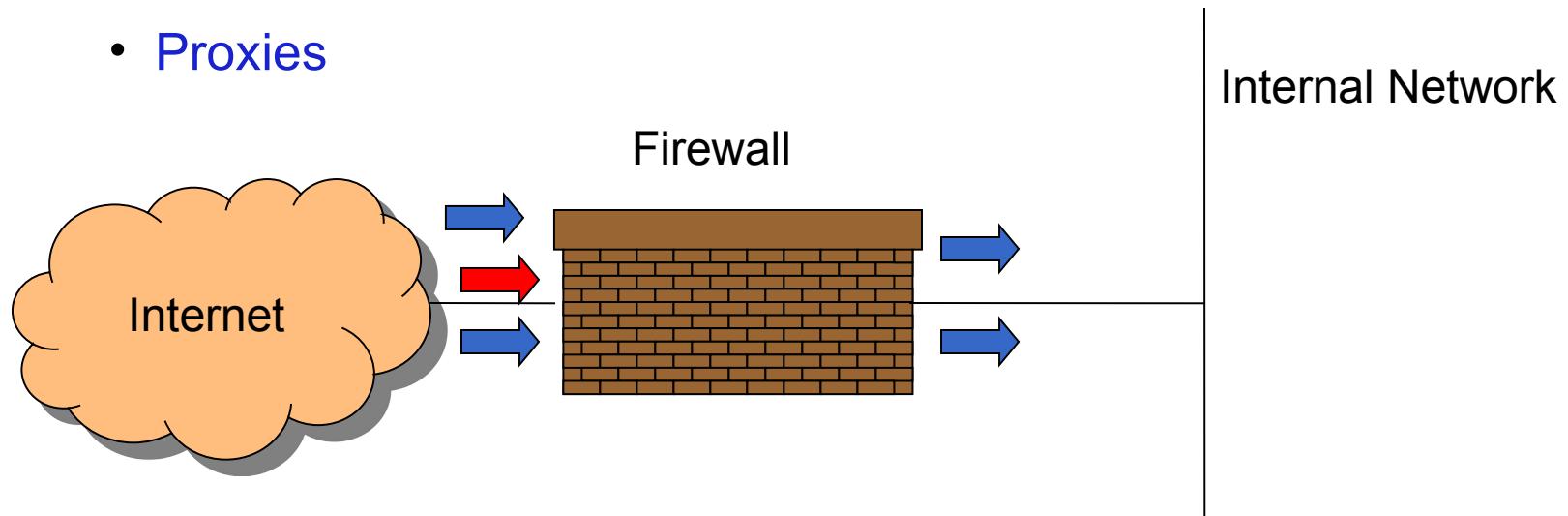
# Firewalls (2)

Firewall actions:

- inspects traffic through it
- allows traffic specified in the policy
- drops everything else

Two Types

- Packet Filters
- Proxies



# I. Packet Filters

- Selectively passes packets from one network interface to another
- Usually done within a router between external and internal network
- What/How to filter?

## Packet Header Fields

- IP source and destination addresses
- Application port numbers
- ICMP message types/ Protocol options etc.

## Packet contents (payloads)

# Packet Filters: Possible Actions

- Allow the packet to go through
- Drop the packet ([Notify Sender/Drop Silently](#))
- Alter the packet ([NAT?](#))
- Log information about the packet

# Packet Filters: Possible Actions

- Allow the packet to go through
- Drop the packet ([Notify Sender/Drop Silently](#))
- Alter the packet ([NAT?](#))
- Log information about the packet

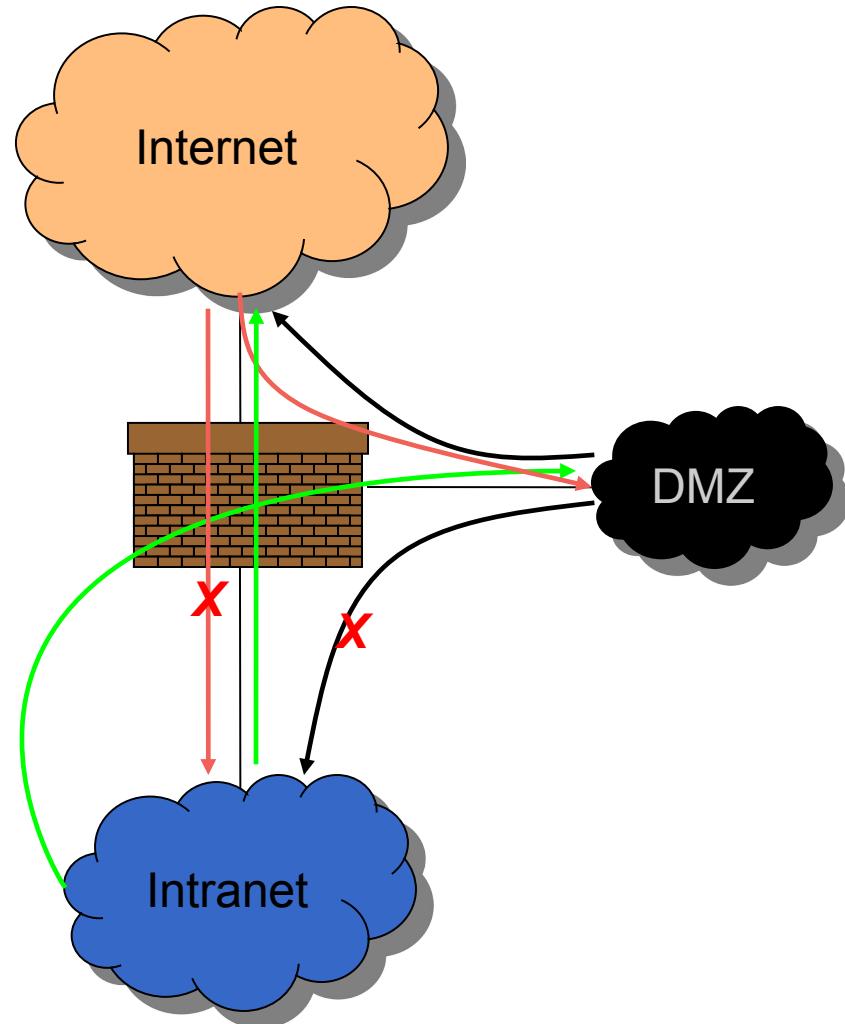
# Some examples

FROM <target a> TO <target b> <action> <protocol> <port>

- Block all packets from outside except for SMTP servers
- Block all traffic to/from a list of domains
- Ingress filtering
  - Drop pkt from outside with addresses inside the network
- Egress filtering
  - Drop pkt from inside with addresses outside the network

# Typical Firewall Configuration

- Internal hosts can access DMZ and Internet
  - External hosts can access DMZ only, not Intranet
  - DMZ hosts can access Internet only
  - **Advantages?**
    - If a service gets compromised in DMZ it cannot affect internal hosts
- \* DMZ - demilitarized zone



# Firewall implementation

Stateless packet filters

Stateful packet filters

# Firewall: Stateless packet filter

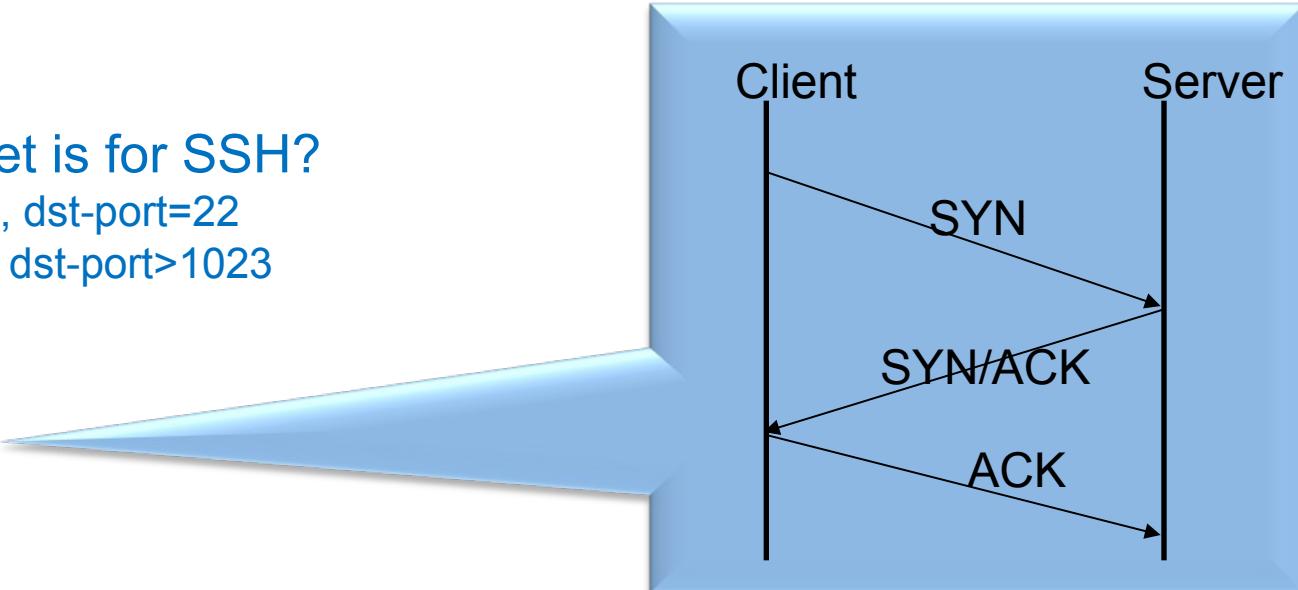
Rule → (Condition, Action)

Rules are processed in top-down order

condition satisfied → action is taken

# Firewall rule example

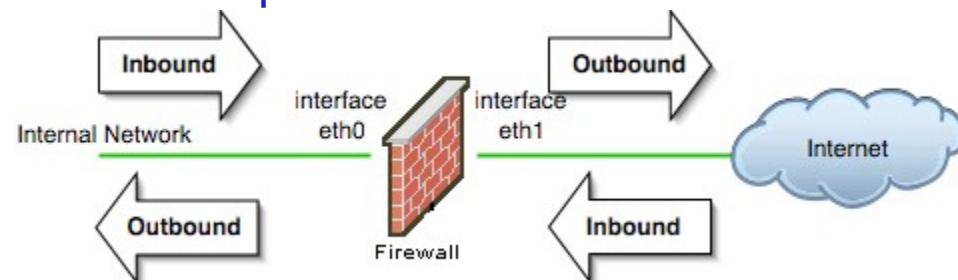
- Allow SSH from external hosts to internal hosts
- Two rules
- Inbound and outbound
- How to know a packet is for SSH?
  - Inbound: src-port>1023, dst-port=22
  - Outbound: src-port=22, dst-port>1023
  - Protocol=TCP
- Ack bit Set?



Rule	Dir	Src Addr	Src Port	Dst Addr	Dst Port	Proto	Ack Set?	Action
SSH-1	In	Ext	> 1023	Int	22	TCP	Any	Allow
SSH-2	Out	Int	22	Ext	> 1023	TCP	Yes	Allow

# Firewall rule example (default)

- Egress Filtering
    - Outbound traffic from external address → Drop
  - Ingress Filtering
    - Inbound Traffic from internal address → Drop
  - Default Deny



# Packet Filters: Pros & Cons

## Advantages

Transparent to application/user

Simple packet filters can be efficient

## Disadvantages

Usually **fail open**

Very hard to configure the rules

Only have coarse-grained information (?)

- Consider these:
  - Does port 22 always mean SSH?
  - Who is the user accessing the SSH?

# Alternatives to Stateless packet filter

- **Stateful packet filters**

Keep the connection states

Easier to specify rules

*Problems?*

- State explosion problem
  - State for TCP/UDP/ICMP? ...

- **Proxy Firewalls**

Two connections instead of one:

- SRV to the client & CLI to the destination server

Either at transport level

- SOCKS proxy

Or at application level

- HTTP proxy

## II. Proxy Firewall

### Data Available

Application level information

User information

### Advantages?

Better policy enforcement

Better logging

### Fail closed

- Shut-down when failure detected

### Disadvantages?

Doesn't perform as well

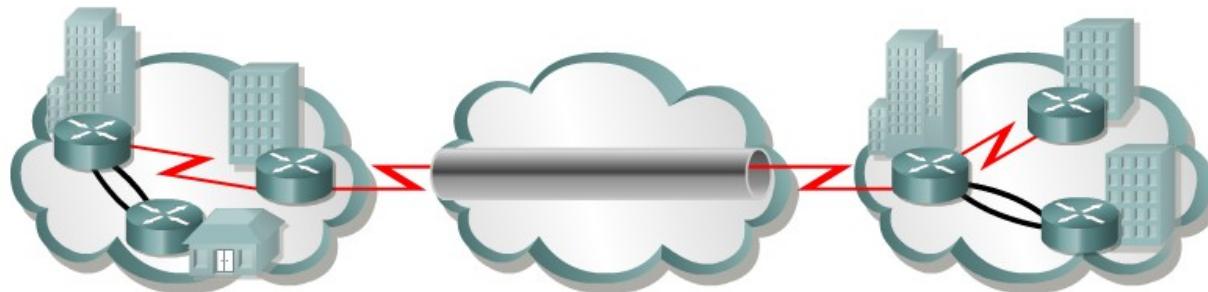
One proxy for each application

Client modification

# VPN

# Virtual Private Networks (VPNs)

- Virtual Private Network (VPN) is defined as network connectivity deployed on a shared infrastructure with the same policies and security as a private network.
- A VPN carries private traffic over a public network using encryption and tunnels to protect the confidentiality of information, integrity of data and authentication of users
- A VPN can be between two end systems, or it can be between two or more networks.
- A VPN can be built using tunnels and encryption. VPNs can occur at any layer of the OSI protocol stack.
- A VPN is an alternative WAN infrastructure that replaces or augments existing private networks that use shared network lines.



# Virtual Private Networks (VPNs)

VPNs provide three critical functions:

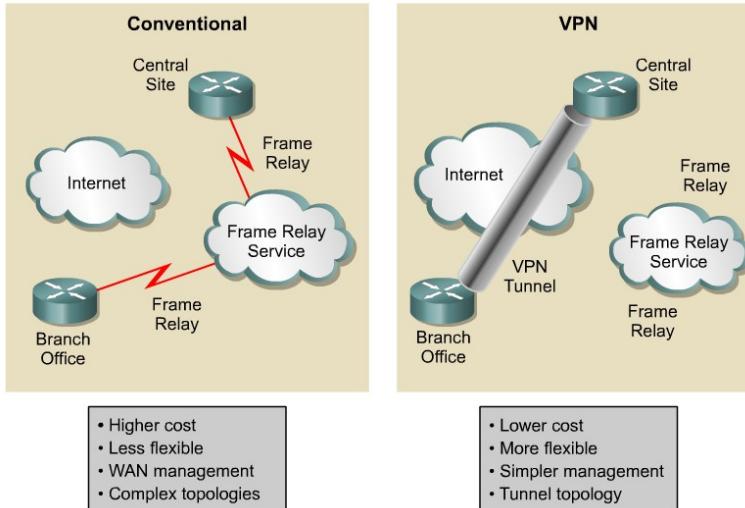
- **Confidentiality (encryption)** – The sender can encrypt the packets before transmitting them across a network. By doing so, no one can access the communication without permission. If intercepted, the communications cannot be read.
- **Data integrity** – The receiver can verify that the data was transmitted through the Internet without being altered.
- **Origin authentication** – The receiver can authenticate the source of the packet, guaranteeing and certifying the source of the information.



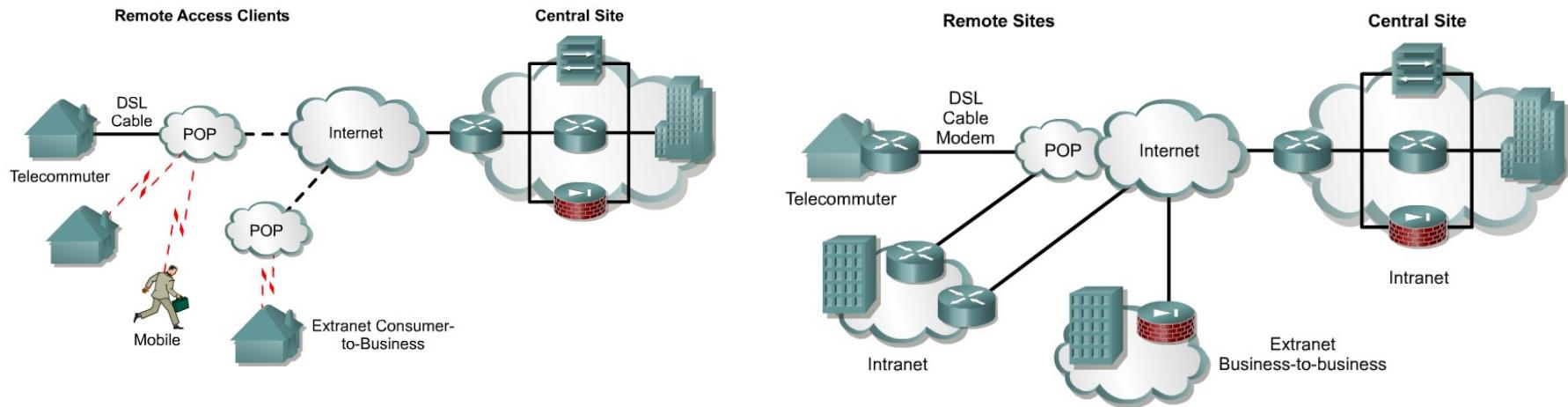
# Virtual Private Networks (VPNs)

The primary benefits include:

- VPNs offer lower cost than private networks. LAN-to-LAN connectivity costs are typically reduced by 20%-40% over domestic leased-line networks.
- VPNs offer flexibility for enabling the Internet economy. VPNs are inherently more flexible and scalable network architectures than classic WANs.
- VPNs provide tunneled network topologies that reduce management burdens.



# VPN usage scenarios



There are two types of remote access VPNs:

- **Client-Initiated** – Remote users use clients to establish a secure tunnel across a shared ISP network to the enterprise.
- **Network Access Server-initiated** – Remote users dial in to an ISP.
  - The Network Access Server establishes a secure tunnel to the enterprise private network that might support multiple remote user-initiated sessions.

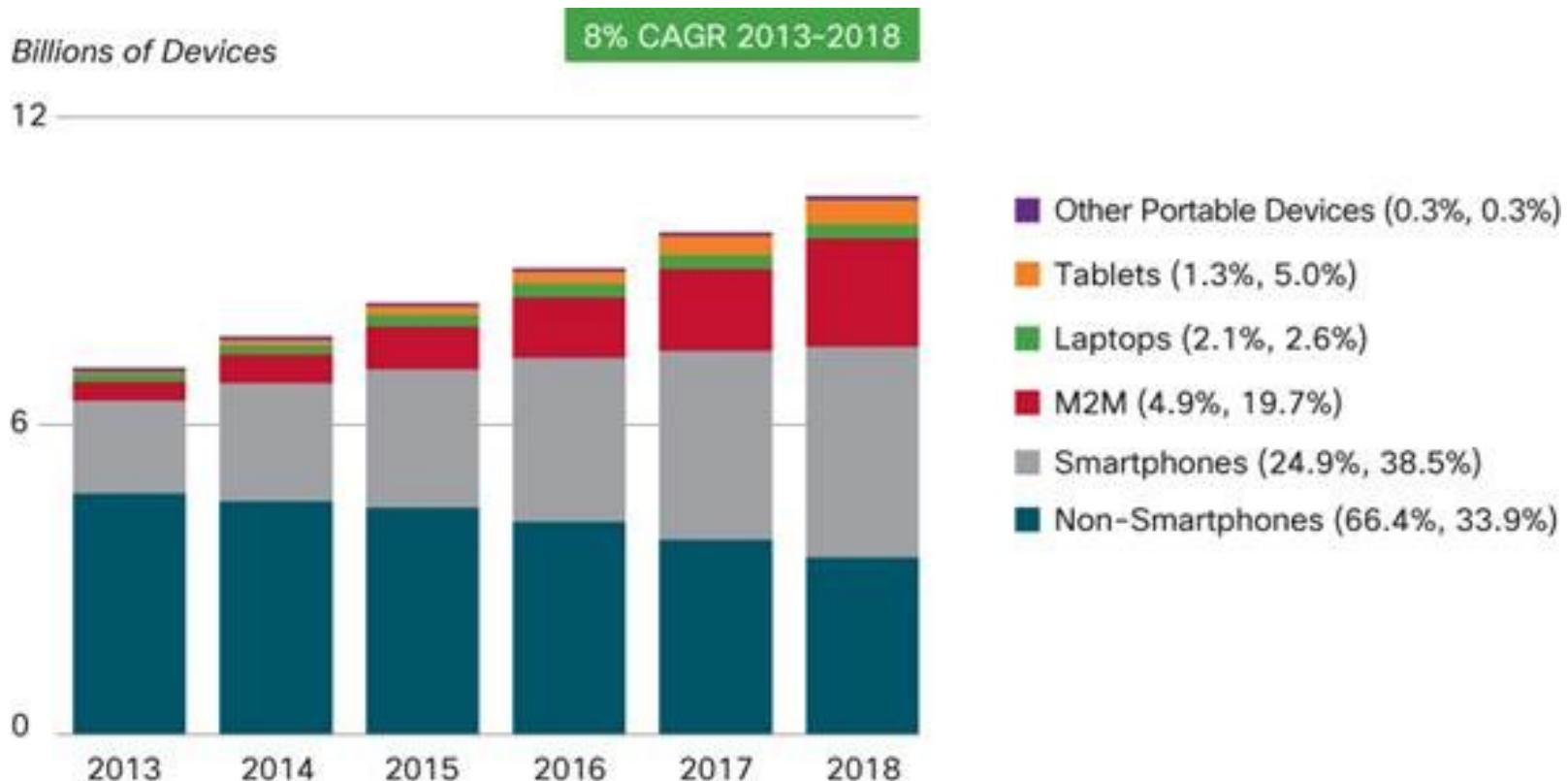
POP = Point of Presence

# WiFi Security

# Wireless basics



# Distribution of devices on the internet



Figures in parentheses refer to device or connections share in 2013, 2018.

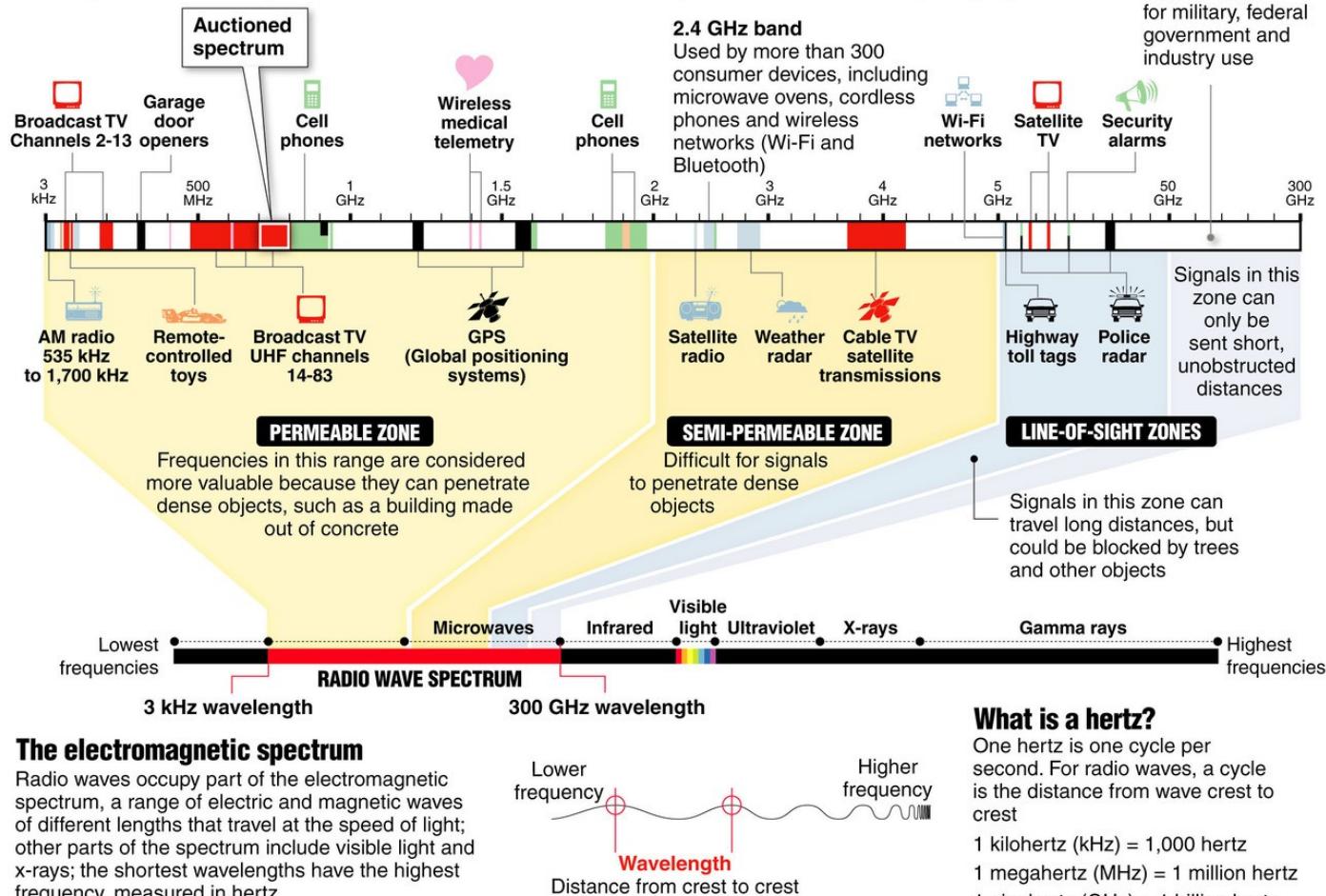
Source: Cisco VNI Mobile, 2014

# Inside the radio wave spectrum

Almost every wireless technology – from cell phones to garage door openers – uses radio waves to communicate.

Some services, such as TV and radio broadcasts, have exclusive use of their frequency within a geographic area.

But many devices share frequencies, which can cause interference. Examples of radio waves used by everyday devices:



## The electromagnetic spectrum

Radio waves occupy part of the electromagnetic spectrum, a range of electric and magnetic waves of different lengths that travel at the speed of light; other parts of the spectrum include visible light and x-rays; the shortest wavelengths have the highest frequency, measured in hertz

Source: New America Foundation, MCT, Howstuffworks.com  
 Graphic: Nathaniel Levine, Sacramento Bee

## What is a hertz?

One hertz is one cycle per second. For radio waves, a cycle is the distance from wave crest to crest

1 kilohertz (kHz) = 1,000 hertz

1 megahertz (MHz) = 1 million hertz

1 gigahertz (GHz) = 1 billion hertz

© 2008 MCT

# Wireless basics

- Mobility
  - laptops,
  - PDA,
  - smartphones,
  - tablets
  - etc
- Easy to install
  - regular rooms
  - conference rooms
  - old buildings
- Cheap ☺

# Technologies

TABLE 1: IEEE 802.11 COMMON WIFI STANDARDS BREAKDOWN

Standard	Frequency Band	Bandwidth	Modulation Scheme	Channel Arch.	Maximum Data Rate	Range	Max Transmit Power
802.11	2.4 GHz	20 MHz	BPSK to 256-QAM	DSSS, FHSS	2 Mbps	20 m	100 mW
b	2.4 GHz	21 MHz	BPSK to 256-QAM	CCK, DSSS	11 Mbps	35 m	100 mW
a	5 GHz	22 MHz	BPSK to 256-QAM	OFDM	54 Mbps	35 m	100 mW
g	2.4 GHz	23 MHz	BPSK to 256-QAM	DSSS, OFDM	54 Mbps	70 m	100 mW
n	2.4 GHz, 5 GHz	24 MHz and 40 MHz	BPSK to 256-QAM	OFDM	600 Mbps	70 m	100 mW
ac	5 GHz	20, 40, 80, 80+80=160 MHz	BPSK to 256-QAM	OFDM	6.93 Gbps	35 m	160 mW
ad	60 GHz	2.16 GHz	BPSK to 64-QAM	SC, OFDM	6.76 Gbps	10 m	10 mW
af	54-790 MHz	6, 7, and 8 MHz	BPSK to 256-QAM	SC, OFDM	26.7 Mbps	>1km ?	100 mW
ah	900 MHz	1, 2, 4, 8, and 16 MHz	BPSK to 256-QAM	SC, OFDM	40 Mbps	1 km	100 mW

# IEEE 802.11

- The first wireless standard (1997) → Legacy
- Featured error correction
- 2.5 / 3.6 / 5 GHz frequency bands
- Transmission speed: 1-2 Mbps



# IEEE 802.11a

- Beginning of 1999
- 5 GHz frequency
- Theoretical speed: 54 Mbps
- Actual speed: 20 Mbps (error correction+overhead+ACK)
- Used expensive equipment

# IEEE 802.11b

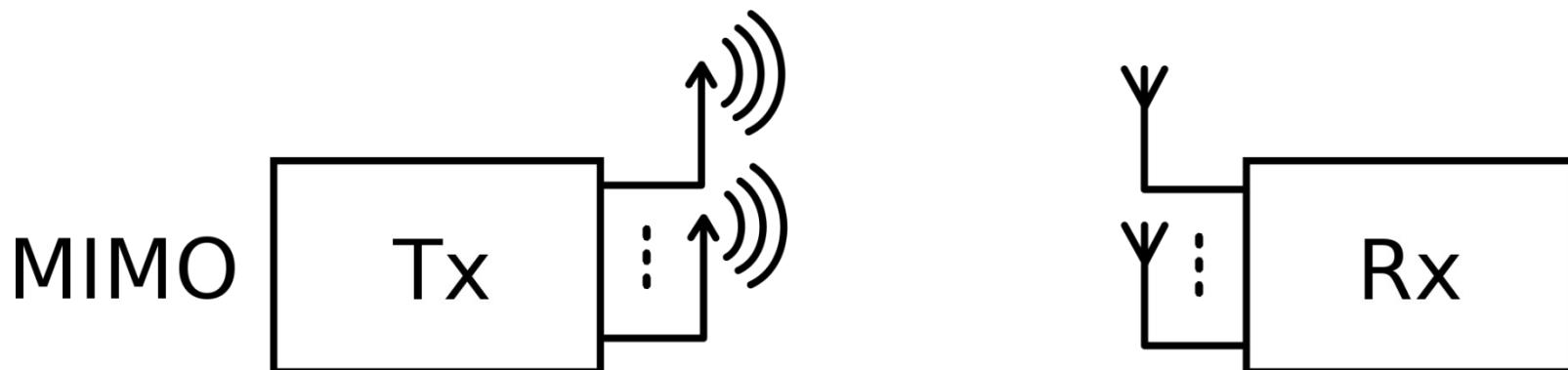
- Mid 1999
- WiFi
- The first popular wireless standard
- Cheaper than 802.11a
- Propagation range: 120m
- 2.4 GHz frequency → a lot of interference
- Theoretical speed: 11 Mbps
- Actual speed: 5 Mbps (error correction+overhead+ACK)

# IEEE 802.11g

- June 2003
- Compatible with 802.11b
- Dominant standard: speed and backwards compatibility
- Propagation range: 100m
- 2.4 GHz frequency → a lot of interference
- Theoretical speed: 54 Mbps
- Actual speed: 22 Mbps (error correction+overhead+ACK)

# IEEE 802.11n

- 29 October 2009
- Compatible with 802.11a/b/g
- Theoretical speed: 600 Mbps
- Propagation range: 250m
- Uses packet aggregation (a single header for multiple data packets)
- Uses multiple antennas with Multiple Input Multiple Output (MIMO) technology



# Wireless security Threats

- **Disclosure threat:** Information is easily leaked to an unwanted party. The transmission medium is by definition Broadcast only.
- **Integrity threat:** There may be unauthorized changes of information during transmission.
- **Denial of service threat:** The transmission frequencies can be easily saturated by outside traffic or white noise.
- There is no centralized, trusted third party for a wireless network

## SSID Broadcast

- Each wireless network has an identifier: SSID (Service Set Identifier) or ESSID (Extended SSID)
- We need to know the SSID to connect to a network
- Access points (AP) broadcast periodically the SSID
- An attacker knowing the SSID can penetrate a network
- **Solution:** block the SSID broadcasting and manually enter the network name

# SSID Broadcast

## The truth:

- **It's a useless security measure!**
- 802.11 protocol defines **beacon** messages
- AP use this to check if a station is still active
- Beacons are seen by everybody and contain the SSID
- An attacker finds the SSID from the beacons

# MAC Address Filtering

- Commonly used in small networks
- Configure the AP to permit access only to a defined list of MAC addresses

## The truth:

- **It's also a useless security measure!**
- An attacker can intercept the traffic from the client and AP, find the MAC address and spoof it

# WEP Security

- WEP = Wired Equivalent Privacy
- Introduced in 1999
- Key length: 10 or 26 hexadecimal digits (40-bit or 104-bit)
- Uses the symmetric RC4 encryption algorithm with a static initialization vector (IV)

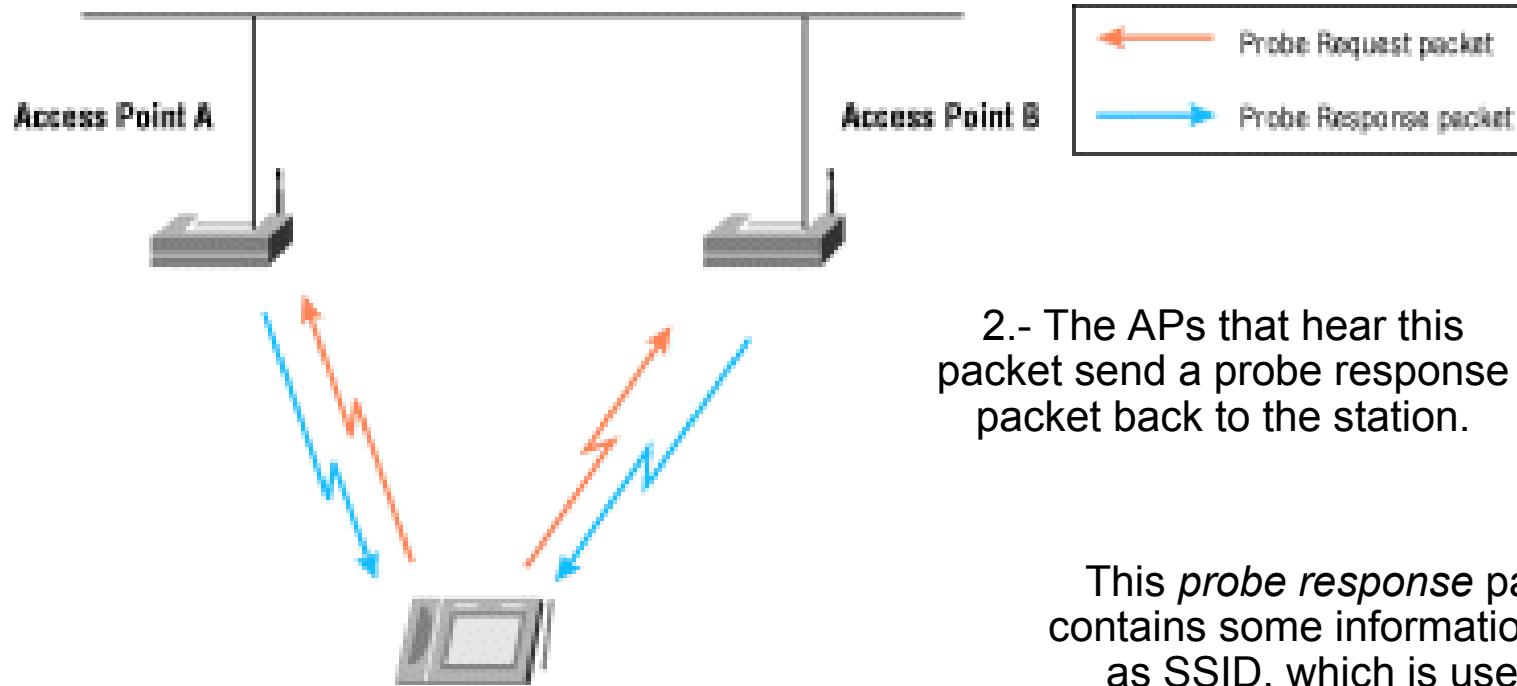
# WEP Security

## How it works:

- a secret key (not the SSID) is known by all the clients
- when a new client (C) is connecting, the AP checks if C knows the secret key → **challenge request**
  - the AP sends a random message and C returns it encrypted.
  - AP decrypts it and compares it with the original message

**Vulnerability:** the AP uses the same key for all traffic;  
the attacker knows the IV of every packet

# WEP Security

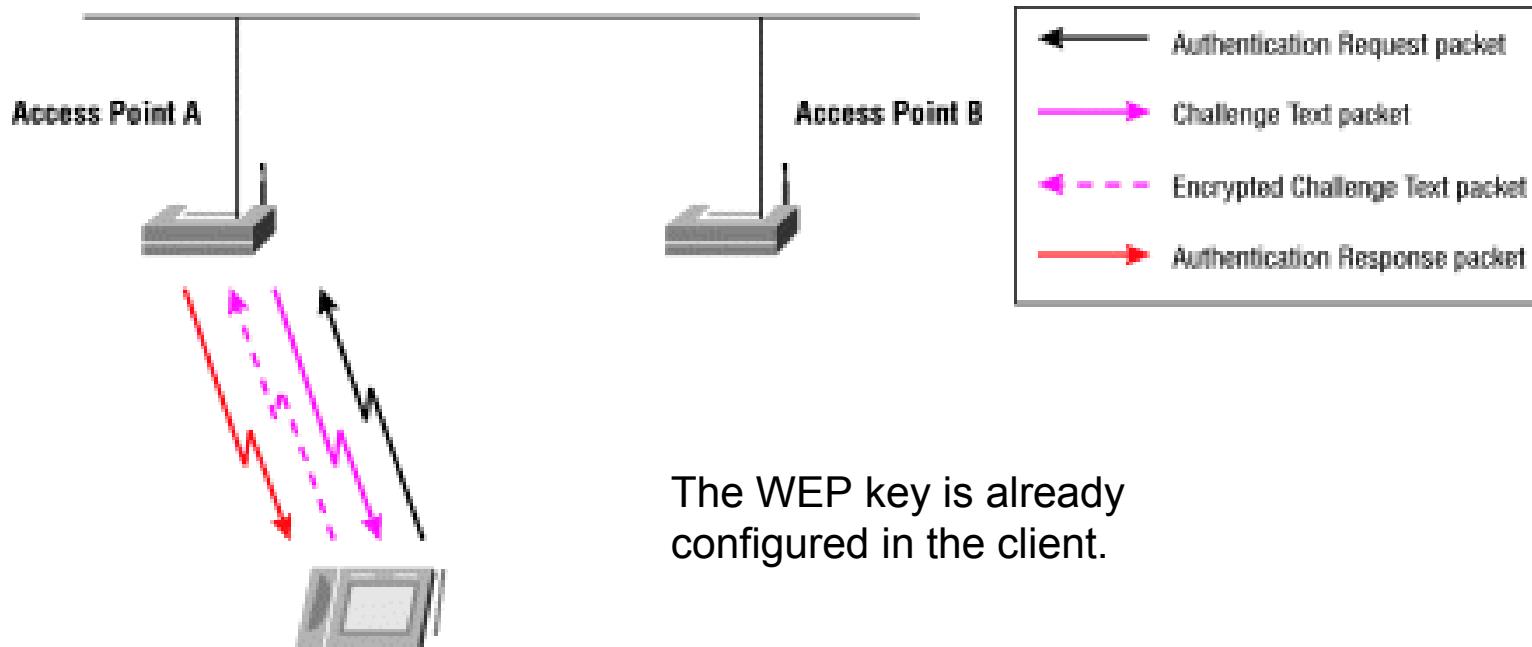


1.- When initialization, the client sends a *probe request* packet out on all the channels.

2.- The APs that hear this packet send a probe response packet back to the station.

This *probe response* packet contains some information such as SSID, which is used to determine which AP associate.

# WEP Security



# WEP Cracking

- **aircrack-ng**
- **aircrack-ng** can be found preinstalled in the BackTrack Linux distribution
- Requirements:
  - A wireless network card\* ☺
  - MAC address of the AP (the BSSID – Basic SSID)
  - Network ESSID
  - AP channel

# WEP Cracking

- Steps:

Start the wireless card in monitor mode

Find the AP name

Start listening for packets (IVs) and save them into a file using ***airodump-ng***

Authenticate with the AP using ***aireplay-ng***

If not enough clients connected to the network, inject fake requests using ***aireplay-ng***

Analyze the file and crack the password using ***aircrack-ng***

# WPA/WPA2 Security

- WPA= Wi-Fi Protected Access
- Uses TKIP (Temporary Key Integrity Protocol)
  - 128 bit
  - Dynamic protocol for changing the encryption key
  - Uses a different key for every packet
- 2 variants
  - Personal WPA (using a pre-shared key)
  - Enterprise WPA (using a RADIUS server)

# WPA/WPA2 Security

- WPA2 uses AES encryption
  - Uses CCMP for data encryption (Counter Cipher Mode with Block Chaining Message Authentication Code Protocol)
  - CCMP is safer than TKIP
- WPA2 cannot be cracked by a non-associated client.....Sort of....See next slide

# WPA/WPA2 Security

## Krack – Key Reinstallation Attack for WPA2

- Presented on 1<sup>st</sup> November 2017
- Replays handshake messages to force the access point to reuse the same key.
- WPA2 does not guarantee that a key is used only once.

<https://www.krackattacks.com/>

<https://papers.mathyvanhoef.com/ccs2017.pdf>

# WPS Security

- Introduced in 2007
- Allows easy setup of a secure wireless home network
- Allow home users who know little of wireless security and may be intimidated by the available security options to setup a Wi-Fi network
- Can be found in almost every modern home routers/APs

# WPS Security

- Methods:
  - ✓ **PIN** – read from a sticker existing on the device and entered on the client side
  - ✓ **Push-Button** – the user pushes a button from the device and a button on the connecting device
  - ✓ **Near-Field-Communication** – the user is located near the device

# WPS Security

- Easy to crack using brute-force attack
  - The attack was presented in December 2011
  - The first public tool: January 2012 → **reaver**
- The flaw allows a remote attacker to recover the WPS PIN in a few hours (the network WPA/WPA2 pre-shared key is found too)
- Cannot be turned-off on some routers
- The exact details:  
**[http://sviehb.files.wordpress.com/2011/12/viehboeck\\_wps.pdf](http://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf)**

# The End

# Security Protocols

Lecture  
4

# SSL/TLS

Cute quote I found:

*“Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.”*

*Gene Spafford  
Professor, Purdue University*

# SSL/TLS Overview

- SSL = Secure Sockets Layer.
  - unreleased v1, flawed but useful v2, good v3.
- TLS = Transport Layer Security.
  - TLS1.0 = SSL3.0 with minor tweaks (see later).
- Defined in RFC 2246 (<https://tools.ietf.org/html/rfc2246>).
  - Open-source implementation at <http://www.openssl.org/>.
- SSL/TLS provides security ‘at TCP layer’.
- Uses TCP to provide reliable, end-to-end transport.
- Applications need some modification.
- In fact, usually a thin layer between TCP and HTTP.

# SSL Handshake Protocol

## Security Goals

- Entity authentication of participating parties.
  - » Participants are called ‘client’ and ‘server’.
  - » Server nearly always authenticated, client more rarely.
  - » Appropriate for most e-commerce applications.
- Establishment of a fresh, shared secret.
  - » Shared secret used to derive further keys.
  - » For confidentiality and authentication in SSL Record Protocol.
- Secure ciphersuite negotiation.
  - » Encryption and hash algorithms
  - » Authentication and key establishment methods.

# SSL/TLS Basic Features

- SSL is a layered protocol
- SSL takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result
- Received data is decrypted, verified, decompressed, and reassembled, then delivered to higher level clients.
- Connects on port 443 by default
- session-identifier cache timeout value of 100 seconds

# SSL/TLS Basic Features

- Confidentiality
  - Encrypt data being sent between client and server, so that passive wiretappers cannot read sensitive data.
- Integrity Protection
  - Protect against modification of messages by an active wiretapper.
- Authentication
  - Verify that a peer is who they claim to be.  
Servers are usually authenticated, and clients may be authenticated if requested by servers

# SSL/TLS Basic Features

- SSL uses TCP/IP on behalf of the higher-level protocols.
- Allows an SSL-enabled server to authenticate itself to an SSL-enabled client;
- Allows the client to authenticate itself to the server;
- Allows both machines to establish an encrypted connection.
- SSL/TLS widely used in Web browsers and servers to support ‘secure e-commerce’ over HTTP.
  - Built into Microsoft IE, Netscape, Mozilla, Apache, IIS,...
  - The (in)famous browser lock.
- SSL architecture provides two layers:
  - SSL Record Protocol
  - Provides secure, reliable channel to upper layer.
  - Upper layer carrying:
    - SSL Handshake Protocol, Change Cipher Spec. Protocol, Alert Protocol, HTTP, any other application protocols.

# Session Example

Firefox ▾

PostFinance - Home Die Schweizerische Post (CH) https://www.postfinance.ch/en.html net craft certificate report Search

You are connected to **postfinance.ch**  
which is run by **Die Schweizerische Post**  
Bern  
Bern, CH  
Verified by: VeriSign, Inc.

Your connection to this web site is encrypted to prevent eavesdropping.

PostFinance Mobile CHF 25,000 to be won

Private customers Business customers

Everything you need for your private finances.  
→ [Products](#)  
→ [E-finance](#)  
→ [E-trading](#)

Everything you need for your business finances.  
→ [Products](#)  
→ [Company founder and start-up](#)  
→ [Small and medium-sized enterprises](#)  
→ [Medium-sized/large businesses](#)  
→ [Public entities](#)  
→ [Associations](#)  
→ [Banks](#)

Page Info - https://www.postfinance.ch/en.html

General Media Permissions Security

Web Site Identity

Web site: **www.postfinance.ch**  
Owner: **Die Schweizerische Post**  
Verified by: **VeriSign, Inc.**

View Certificate

Privacy & History

Have I visited this web site prior to today? Yes, 6 times

Is this web site storing information (cookies) on my computer? Yes

View Cookies

Have I saved any passwords for this web site? No

View Saved Passwords

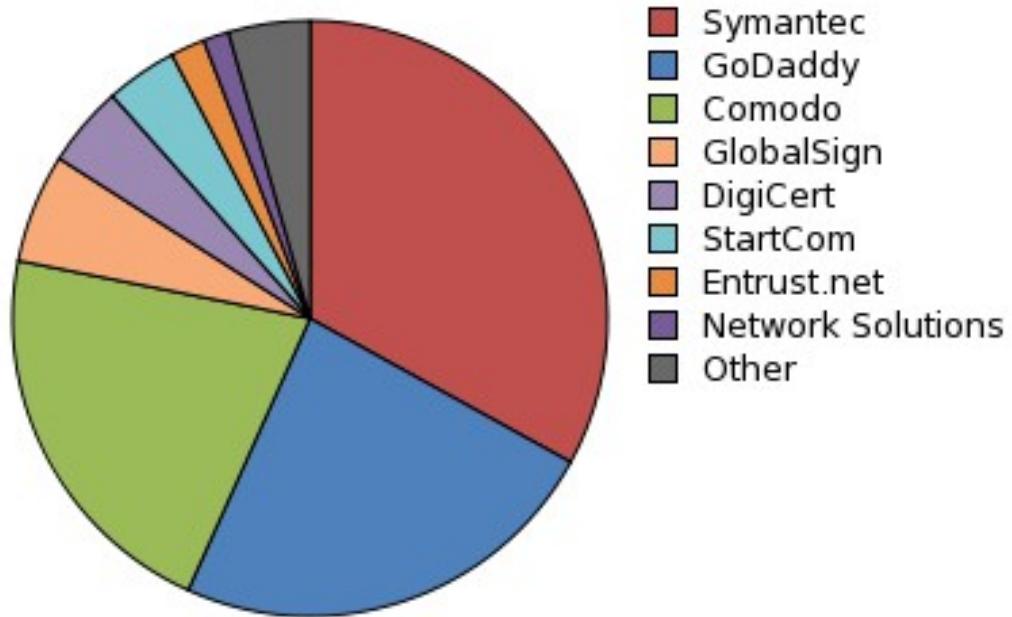
Technical Details

**Connection Encrypted: High-grade Encryption (AES-256, 256 bit keys)**

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.

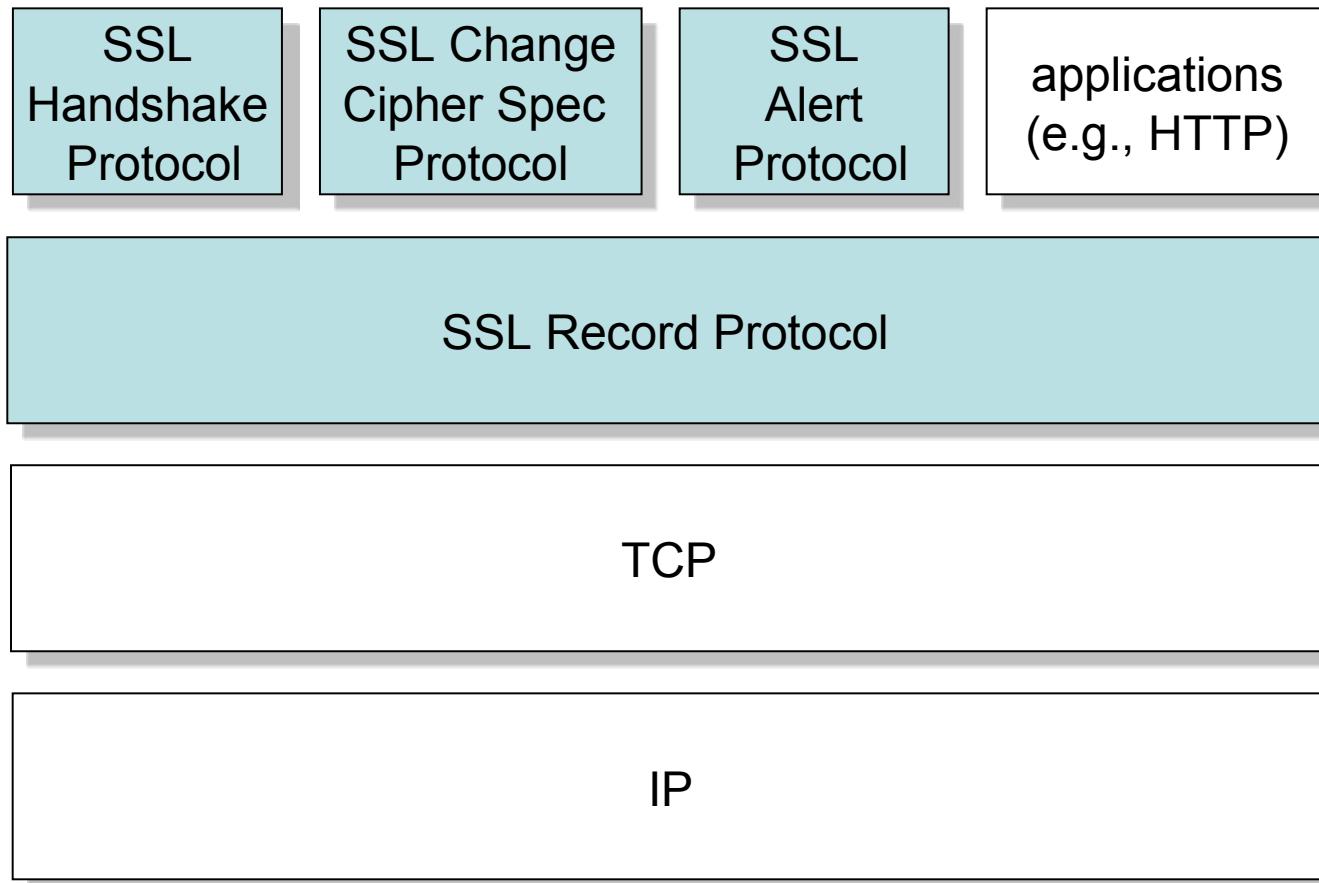
# Market share of certification authorities



2015 Netcraft Ltd

<https://ssl.netcraft.com/ssl-sample-report/CMatch/certs>

# SSL Protocol Architecture



# SSL components

- SSL Handshake Protocol
  - negotiation of security algorithms and parameters
  - key exchange
  - server authentication and optionally client authentication
- SSL Record Protocol
  - fragmentation
  - compression
  - message authentication and integrity protection
  - encryption
- SSL Alert Protocol
  - error messages (fatal alerts and warnings)
- SSL Change Cipher Spec Protocol
  - a single message that indicates the end of the SSL handshake

# Sessions and connections

- an **SSL session** is an association between a client and a server
- sessions are stateful; the session state includes security algorithms and parameters
- a session may include multiple secure connections between the same client and server
- connections of the same session share the session state
- sessions are used to avoid expensive negotiation of new security parameters for each connection
- there may be multiple simultaneous sessions between the same two parties, but this feature is not used in practice

session state

session identifier

- arbitrary byte sequence chosen by the server to identify the session

peer certificate

- X509 certificate of the peer
- may be null

compression method

cipher spec

- bulk data encryption algorithm (e.g., null, DES, 3DES, ...)
- MAC algorithm (e.g., MD5, SHA-1)
- cryptographic attributes (e.g., hash size, IV size, ...)

master secret

- 48-byte secret shared between the client and the server

is resumable

- a flag indicating whether the session can be used to initiate new connections

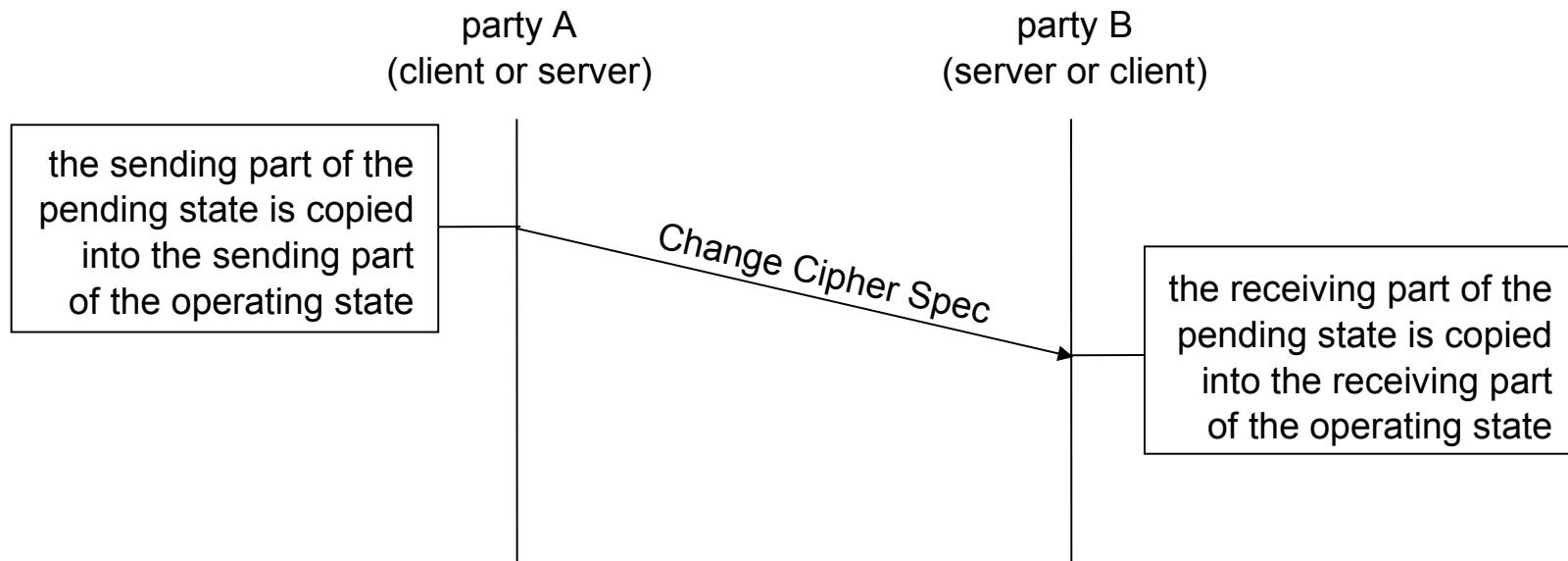
connection states

# Session and connection states (2)

- connection state
  - server and client random
    - random byte sequences chosen by the server and the client for every connection
  - server write MAC secret
    - secret key used in MAC operations on data sent by the server
  - client write MAC secret
    - secret key used in MAC operations on data sent by the client
  - server write key
    - secret encryption key for data encrypted by the server
  - client write key
    - secret encryption key for data encrypted by the client
  - initialization vectors
    - an IV is maintained for each encryption key if CBC mode is used
    - initialized by the SSL Handshake Protocol
    - final ciphertext block from each record is used as IV with the following record
  - sending and receiving sequence numbers
    - sequence numbers are 64 bits long
    - reset to zero after each Change Cipher Spec message

# State changes

- operating state
  - currently used state
- pending state
  - state to be used
  - built using the current state
- operating state  $\leftarrow$  pending state
  - at the transmission and reception of a Change Cipher Spec message



# SSL Record Protocol

- Provides secure, reliable channel to upper layer.
- Carries application data and SSL ‘management’ data.
- Session concept:
  - » Sessions created by handshake protocol.
  - » Defines set of cryptographic parameters (encryption and hash algorithm, master secret, certificates).
  - » Carries multiple *connections* to avoid repeated use of expensive handshake protocol.
- Connection concept:
  - » State defined by nonces, secret keys for MAC and encryption, IVs, sequence numbers.
  - » Keys for many connections derived from single master secret created during handshake protocol.

# SSL Record Protocol

SSL Record Protocol provides:

Data origin authentication and integrity.

- MAC using algorithm similar to HMAC.
- Based on MD-5 or SHA-1 hash algorithms.
- MAC protects 64 bit sequence number for anti-replay.

Confidentiality.

- Bulk encryption using symmetric algorithm.
  - IDEA, RC2-40, DES-40 (exportable), DES, 3DES,...
  - RC4-40 and RC4-128.

Data from application/upper layer SSL protocol partitioned into fragments (max size  $2^{14}$  bytes).

MAC first then pad (if needed), finally encrypt.

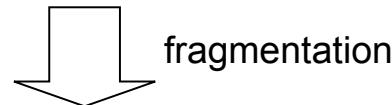
Prepend header.

Content type, version, length of fragment.

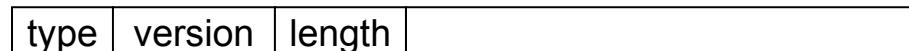
Submit to TCP.

# SSL Record Protocol – processing overview

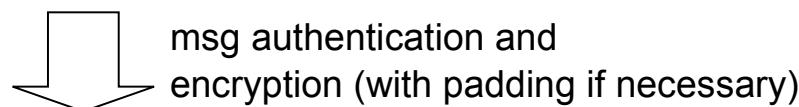
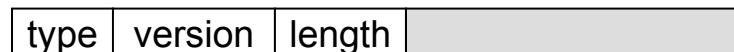
application data



SSLPlaintext



SSLCompressed



SSLCiphertext



$$\text{MAC} = \text{hash}(\text{ MAC\_write\_secret } | \text{pad\_2} | \text{hash}(\text{ MAC\_write\_secret } | \text{pad\_1} | \text{seq\_num} | \text{type} | \text{length} | \text{fragment}))$$

- similar to HMAC but the pads are concatenated
- supported hash functions:
  - MD5
  - SHA-1
  - ....
- pad\_1 is 0x36 repeated 48 times (MD5) or 40 times (SHA-1)
- pad\_2 is 0x5C repeated 48 times (MD5) or 40 times (SHA-1)

# Encryption

- supported algorithms
  - block ciphers (in CBC mode)
    - RC2\_40
    - DES\_40
    - DES\_56
    - 3DES\_168
    - IDEA\_128
    - Fortezza\_80
  - stream ciphers
    - RC4\_40
    - RC4\_128
- if a block cipher is used, then padding is applied
  - last byte of the padding is the padding length

# SSL Alert Protocol

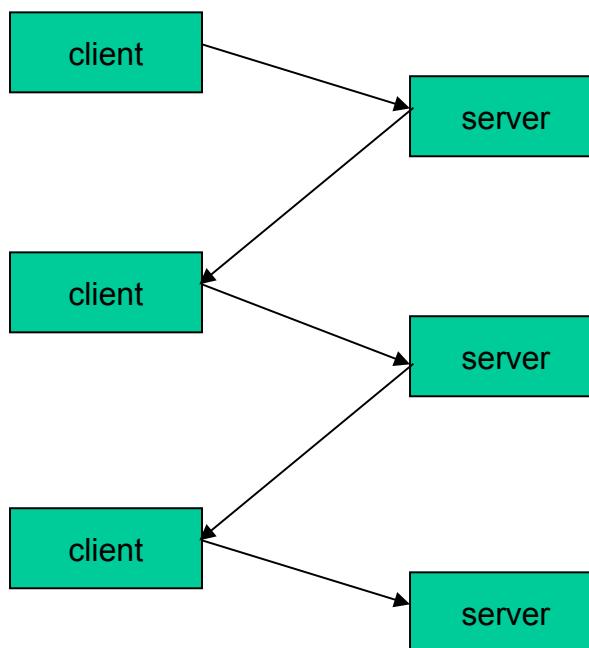
- each alert message consists of 2 fields (bytes)
- first field (byte): “warning” or “fatal”
- second field (byte):
  - fatal
    - unexpected\_message
    - bad\_record\_MAC
    - decompression\_failure
    - handshake\_failure
    - illegal\_parameter
  - warning
    - close\_notify
    - no\_certificate
    - bad\_certificate
    - unsupported\_certificate
    - certificate\_revoked
    - certificate\_expired
    - certificate\_unknown
- in case of a fatal alert
  - connection is terminated
  - session ID is invalidated → no new connection can be established within this session

# SSL Handshake Protocol – overview

Client send hello message including a random message and its protocol version, session ID, cipher suite, and compression method

Client authenticates server, then creates a pre-master secret for the session and encrypts the message with the servers public key (may send its certificate also)

Client decodes the master key and tells the server that it will use the key to encode the session also.

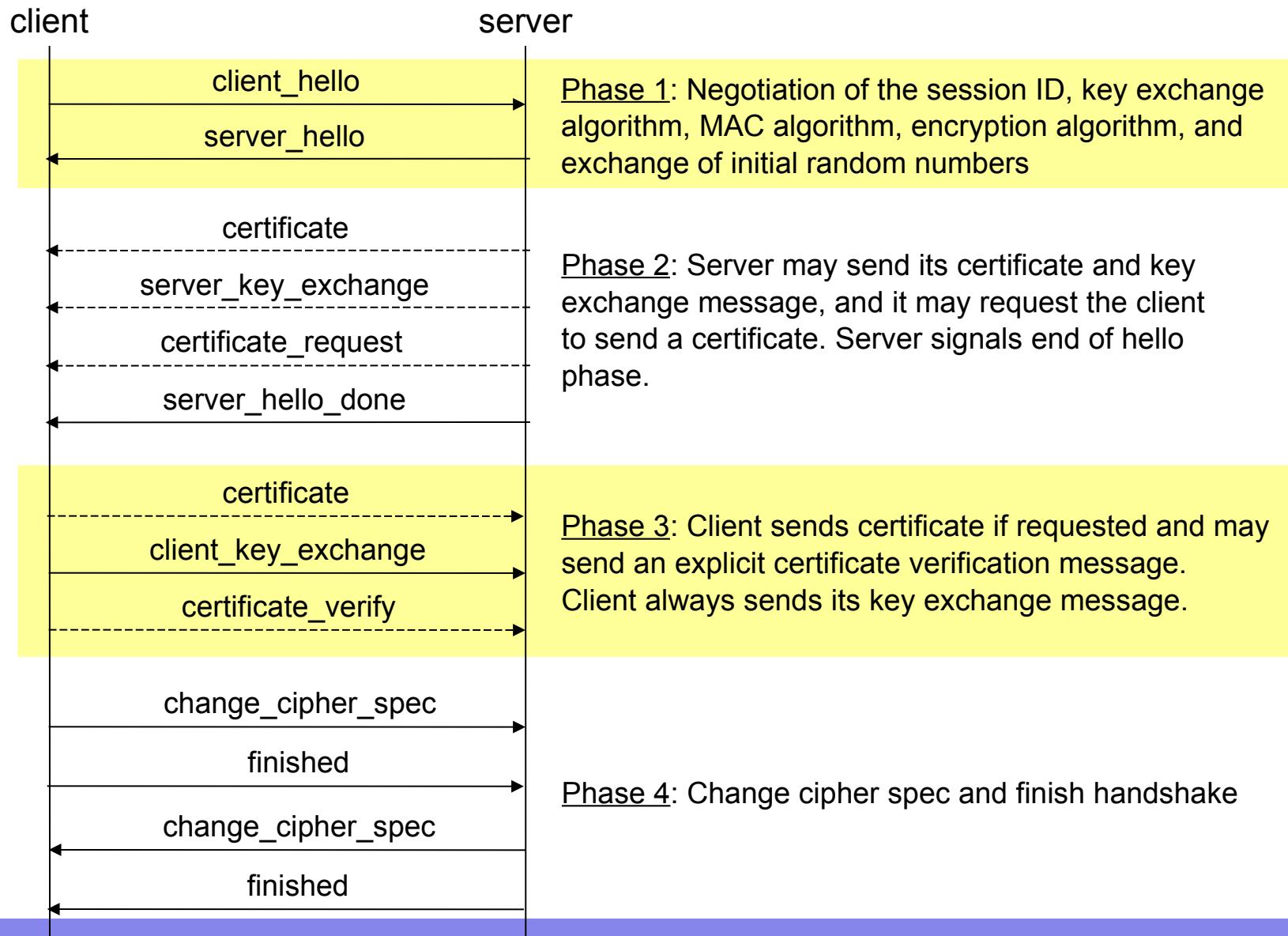


Server replies with a hello message with its own protocols, random message, its certificate and requests for client certificate if necessary

Server authenticates the client if necessary, and uses its private key to decode the message and the pre-master secret, then creates a master secret key for the session and tells the client that it will use the master key for the session

Handshake is done

# SSL Handshake Protocol – overview



# Hello messages

## client\_hello

### client\_version

- the highest version supported by the client

### client\_random

- current time (4 bytes) + pseudo random bytes (28 bytes)

### session\_id

- empty if the client wants to create a new session, or
- the session ID of an old session within which the client wants to create the new connection

### cipher\_suites

- list of cryptographic options supported by the client ordered by preference
- a cipher suite contains the specification of the
  - key exchange method, the encryption and the MAC algorithm
  - the algorithms implicitly specify the hash\_size, IV\_size, and key\_material parameters (part of the Cipher Spec of the session state)
- example: SSL\_RSA\_with\_3DES\_EDE\_CBC\_SHA

### compression\_methods

- list of compression methods supported by the client

# Hello messages

## server\_hello

### server\_version

- min( highest version supported by client, highest version supported by server )

### server\_random

- current time + random bytes
- random bytes must be independent of the client random

### session\_id

- session ID chosen by the server
- if the client wanted to resume an old session:
  - server checks if the session is resumable
  - if so, it responds with the session ID and the parties proceed to the finished messages
- if the client wanted a new session
  - server generates a new session ID

### cipher\_suite

- single cipher suite selected by the server from the list given by the client

### compression\_method

- single compression method selected by the server

# Supported key exchange methods

- RSA based (SSL\_RSA\_with...)
  - the secret key (pre-master secret) is encrypted with the server's public RSA key
  - The server's public key is made available to the client during the exchange
- fixed Diffie-Hellman (SSL\_DH\_RSA\_with... or SSL\_DH\_DSS\_with...)
  - the server has fix DH parameters contained in a certificate signed by a CA
  - the client may have fix DH parameters certified by a CA or it may send an unauthenticated one-time DH public value in the client\_key\_exchange message
- ephemeral Diffie-Hellman (SSL\_DHE\_RSA\_with... or SSL\_DHE\_DSS\_with...)
  - both the server and the client generate one-time DH parameters
  - the server signs its DH parameters with its private RSA or DSS key
  - the client may authenticate itself (if requested by the server) by signing the hash of the handshake messages with its private RSA or DSS key
- anonymous Diffie-Hellman
  - both the server and the client generate one-time DH parameters
  - they send their parameters to the peer without authentication
- Fortezza
  - Fortezza proprietary key exchange scheme

# Server certificate and key exchange messages

- certificate
  - required for every key exchange method except for anonymous DH
  - contains one or a chain of X.509 certificates (up to a known root CA)
  - may contain
    - public RSA key suitable for encryption, or
    - public RSA or DSS key suitable for signing only, or
    - fix DH parameters
- server\_key\_exchange
  - sent only if the certificate does not contain enough information to complete the key exchange (e.g., the certificate contains an RSA signing key only)
  - may contain
    - public RSA key (exponent and modulus), or
    - DH parameters ( $p$ ,  $g$ , public DH value), or
    - Fortezza parameters
  - Digitally signed
    - if DSS: SHA-1 hash of (client\_random | server\_random | server\_params) is signed
    - if RSA: MD5 hash and SHA-1 hash of (client\_random | server\_random | server\_params) are concatenated and encrypted with the private RSA key

# Certificate request and server hello done messages

- certificate\_request
  - sent if the client needs to authenticate itself
  - specifies which type of certificate is requested (rsa\_sign, dss\_sign, rsa\_fixed\_dh, dss\_fixed\_dh, ...)
- server\_hello\_done
  - sent to indicate that the server is finished its part of the key exchange
  - after sending this message the server waits for client response
  - the client should verify that the server provided a valid certificate and the server parameters are acceptable

# Client authentication and key exchange

- certificate
    - sent only if requested by the server
    - may contain
      - public RSA or DSS key suitable for signing only, or
      - fix DH parameters
  - client\_key\_exchange
    - always sent (but it is empty if the key exchange method is fix DH)
    - may contain
      - RSA encrypted pre-master secret, or
      - client one-time public DH value, or
      - Fortezza key exchange parameters
  - certificate\_verify
    - sent only if the client sent a certificate
    - provides client authentication
    - contains signed hash of all the previous handshake messages
      - if DSS: SHA-1 hash is signed
      - if RSA: MD5 and SHA-1 hash is concatenated and encrypted with the private key
- MD5( master\_secret | pad\_2 | MD5( handshake\_messages | master\_secret | pad\_1 ) )  
SHA( master\_secret | pad\_2 | SHA( handshake\_messages | master\_secret | pad\_1 ) )

# Finished messages

- finished
  - sent immediately after the change\_cipher\_spec message
  - first message that uses the newly negotiated algorithms, keys, IVs, etc.
  - used to verify that the key exchange and authentication was successful
  - contains the MD5 and SHA-1 hash of all the previous handshake messages:

$\text{MD5}(\text{master\_secret} \mid \text{pad\_2} \mid \text{MD5}(\text{handshake\_messages} \mid \text{sender} \mid \text{master\_secret} \mid \text{pad\_1})) \mid \text{SHA}(\text{master\_secret} \mid \text{pad\_2} \mid \text{SHA}(\text{handshake\_messages} \mid \text{sender} \mid \text{master\_secret} \mid \text{pad\_1}))$

where “sender” is a code that identifies that the sender is the client or the server (client: 0x434C4E54; server: 0x53525652)

# Cryptographic computations

- pre-master secret
  - if key exchange is RSA based:
    - generated by the client
    - sent to the server encrypted with the server's public RSA key
  - if key exchange is Diffie-Hellman based:
    - $\text{pre\_master\_secret} = g^{xy} \bmod p$

- master secret (48 bytes)

```
master_secret = MD5( pre_master_secret | SHA( "A" | pre_master_secret | client_random | server_random ) ) |
               MD5( pre_master_secret | SHA( "BB" | pre_master_secret | client_random | server_random ) ) |
               MD5( pre_master_secret | SHA( "CCC" | pre_master_secret | client_random | server_random ) )
```

- keys, MAC secrets, IVs

```
MD5( master_secret | SHA( "A" | master_secret | client_random | server_random ) ) |
MD5( master_secret | SHA( "BB" | master_secret | client_random | server_random ) ) |
MD5( master_secret | SHA( "CCC" | master_secret | client_random | server_random ) ) | ...
```



key block :

client write MAC secret	server write MAC secret	client write key	server write key	...
-------------------------	-------------------------	------------------	------------------	-----

# TLS vs. SSL

- TLS1.0 = SSL3.0 with minor differences.
  - TLS signalled by version number 3.1.
- MAC
  - TLS uses HMAC
  - the MAC covers the version field of the record header too
- more alert codes
- cipher suites
  - TLS doesn't support Fortezza key exchange and Fortezza encryption
- certificate\_verify message
  - the hash is computed only over the handshake messages
  - in SSL the hash contained the master\_secret and pads

# TLS vs. SSL

- pseudorandom function PRF

$$\begin{aligned} P\_hash(secret, seed) = & \text{HMAC\_hash( secret, } A(1) \mid seed ) \mid \\ & \text{HMAC\_hash( secret, } A(2) \mid seed ) \mid \\ & \text{HMAC\_hash( secret, } A(3) \mid seed ) \mid \dots \end{aligned}$$

where

$$A(0) = seed$$

$$A(i) = \text{HMAC\_hash}(secret, A(i-1))$$

$$\begin{aligned} \text{PRF}(secret, label, seed) = & P\_\text{MD5}(secret\_left, label \mid seed) \oplus \\ & P\_\text{SHA}(secret\_right, label \mid seed) \end{aligned}$$

# TLS vs. SSL

- finished message
  - PRF( master\_secret,  
“client finished”,  
MD5(handshake\_messages) | SHA(handshake\_messages) )
- cryptographic computations
  - pre-master secret is calculated in the same way as in SSL
  - master secret:  
PRF( pre\_master\_secret,  
“master secret”,  
client\_random | server\_random )
  - key block:  
PRF( master\_secret,  
“key expansion”,  
server\_random | client\_random )
- padding before block cipher encryption
  - variable length padding is allowed (max 255 padding bytes)

# Some SSL/TLS Security Flaws

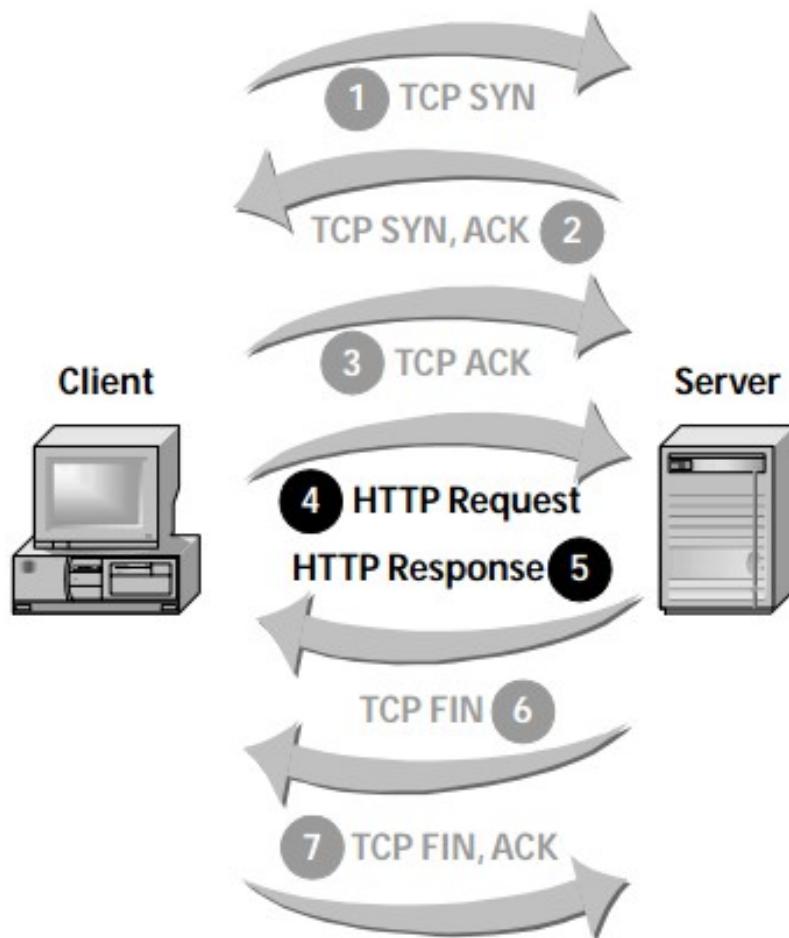
- (Historical) flaws in random number generation for SSL.
  - Low quality RNG leads to predictable session keys.
  - Goldberg and Wagner, Dr. Dobb's Journal, Jan. 1996.
  - <http://www.ddj.com/documents/s=965/ddj9601h/>
- Flaws in error reporting.
  - (differing response times by server in event of padding failure and MAC failure) + (analysis of padding method for CBC-mode) = recovery of SSL plaintext.
  - Canvel, Hiltgen, Vaudenay and Vuagnoux, Crypto2003.
  - [http://lasecwww.epfl.ch/php\\_code/publications/search.php?ref=CHVV03](http://lasecwww.epfl.ch/php_code/publications/search.php?ref=CHVV03)
- Timing attacks.
  - analysis of OpenSSL server response times allows attacker in same LAN segment to derive server's private key!
  - Boneh and Brumley, 12th Usenix Security Symposium, 2003.
  - <http://crypto.stanford.edu/~dabo/abstracts/ssl-timing.html>

**HTTPS = SSL + HTTP**  
**Other applications for SSL**

# An introduction to HTTP

- ▶ Hyper Text Transfer Protocol
- ▶ One of the application layer protocols that make up the Internet
  - HTTP over TCP/IP
  - Like SMTP, POP, IMAP, NNTP, FTP, etc.
- ▶ The underlying language of the Web
- ▶ Three versions have been used, two are in common use and have been specified:
  - RFC 1945 HTTP 1.0 (1996)
  - RFC 2616 HTTP 1.1 (1999)

# HTTP requires a TCP connection



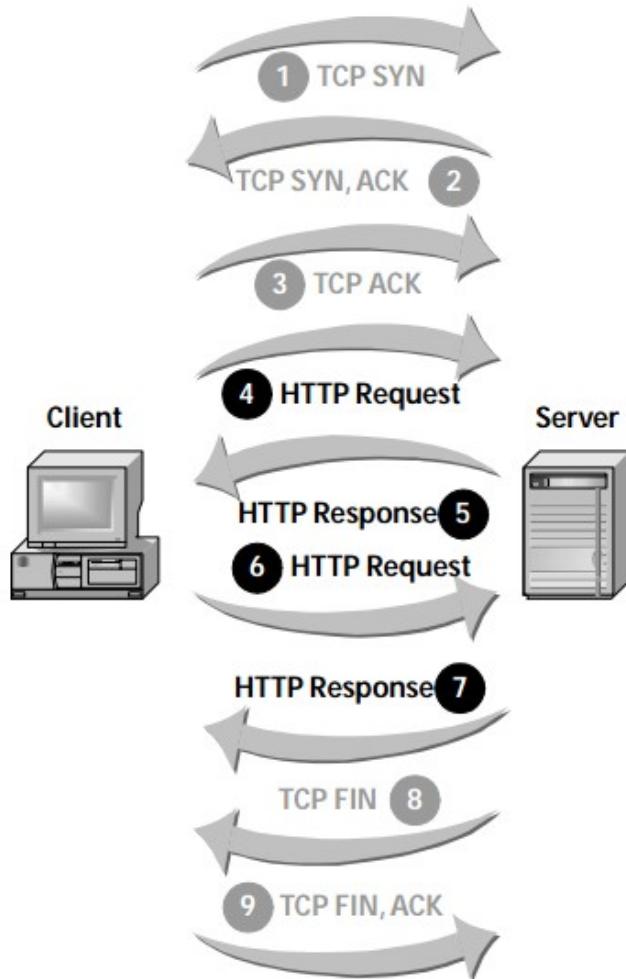
## ◀ Figure

Before systems can exchange HTTP messages, they must establish a TCP connection. Steps 1, 2, and 3 in this example show the connection establishment. Once the TCP connection is available, the client sends the server an HTTP request. The final two steps, 6 and 7, show the closing of the TCP connection.

# Connection Persistence

- ▶ The first versions of HTTP required clients to establish a separate TCP connection with each request.
- ▶ HTTP 1.1 protocol eliminates the problem of multiple TCP connections with a feature known as **persistence**.

# Connection Persistence



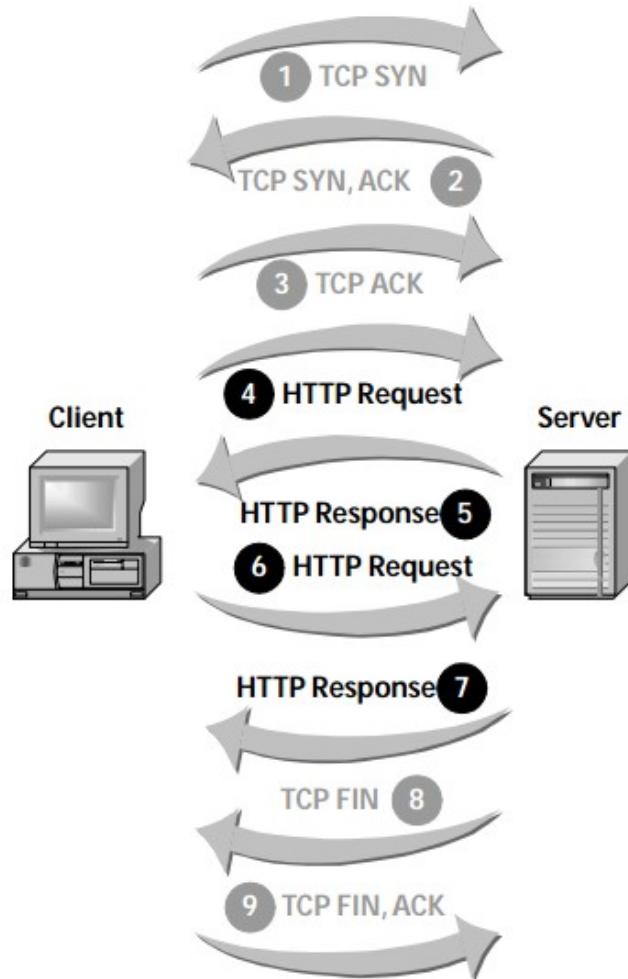
◀ Figure 2.4

With persistent connections, a client can issue many HTTP requests over a single TCP connection. The first request is in step 4, which the server answers in step 5. In step 6 the client continues by sending the server another request on the same TCP connection. The server responds to this request in step 7 and then closes the TCP connection.

# Pipelining

- ▶ Persistence allows another http feature that improves performance **pipelining**.
- ▶ Pipelining, a client does not have to wait for a response to one request before issuing a new request on the connection.

# Pipelining



◀ Figure 2.4

With persistent connections, a client can issue many HTTP requests over a single TCP connection. The first request is in step 4, which the server answers in step 5. In step 6 the client continues by sending the server another request on the same TCP connection. The server responds to this request in step 7 and then closes the TCP connection.

# HTTPS

- Hypertext Transfer Protocol over Secure Socket Layer (HTTPS)
- HTTPS is the use of Secure Sockets Layer (SSL) as a sub-layer under the regular HTTP in the application layer. It is also referred to as Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) or HTTP over SSL, in short.
- HTTPS is a Web protocol developed by Netscape, and it is built into its browser to encrypt and decrypt user page requests as well as the pages that are returned by the Web server. HTTPS uses port 443 instead of HTTP port 80 in its interactions with the lower layer, TCP/IP.

# SSL/TLS Applications

- Secure e-commerce using SSL/TLS.
  - Client authentication not needed until client decides to buy something.
  - SSL provides secure channel for sending credit card information, personal details, etc.
  - Client authenticated using credit card information, merchant bears (most of) risk.
  - Very successful (amazon.com, on-line supermarkets, airlines,...)

# SSL/TLS Applications

- Secure e-commerce: some issues.
  - No guarantees about what happens to client data (including credit card details) after session: may be stored on insecure server.
  - Does client understand meaning of certificate expiry and other security warnings?
  - Does client software *actually* check complete certificate chain?
  - Does the name in certificate match the URL of e-commerce site? Does the user check this?
  - Is the site the one the client thinks it is?
  - Is the client software proposing appropriate ciphersuites?

# SSL/TLS Applications

- Secure electronic banking.
  - Client authentication may be enabled using client certificates.
  - Issues of registration, secure storage of private keys, revocation and re-issue.
  - Otherwise, SSL provides secure channel for sending username, password, mother's maiden name,...
    - *What else does client use same password for?*
  - Does client understand meaning of certificate expiry and other security warnings?
  - Is client software proposing appropriate ciphersuites?
    - *Enforce from server.*

# The End

# Security Protocols

Lecture  
5

# SSH

# SSH Overview

- SSH = Secure Shell.
  - Initially designed to replace insecure rsh, telnet utilities.
  - Secure remote administration (mostly of Unix systems).
  - Extended to support secure file transfer and e-mail.
  - Latter, provided a general secure channel for network applications.
  - SSH-1 flawed, SSH-2 better security (and different architecture).
- SSH provides security at Application layer.
  - Only covers traffic explicitly protected.
  - Applications need modification, but port-forwarding eases some of this (see later).
  - Built on top of TCP, reliable transport layer protocol.

# SSH-2 Architecture

SSH-2 adopts a three layer architecture:

- SSH Transport Layer Protocol.
  - Initial connection.
  - Server authentication (almost always).
  - Sets up secure channel between client and server.
- SSH Authentication Protocol
  - Client authentication over secure transport layer channel.
- SSH Connection Protocol
  - Supports multiple connections over a single transport layer protocol secure channel.
  - Efficiency (session re-use).

# SSH-2 Architecture

Applications

SSH Connection Protocol

SSH Authentication Protocol

SSH Transport Layer Protocol

TCP

# SSH-2 Security Goals

- Server (nearly) always authenticated in transport layer protocol.
- Client (nearly) always authenticated in authentication protocol.
  - By public key (DSS, RSA, SPKI, OpenPGP).
  - Or simple password for particular application over secure channel.
- Establishment of a fresh, shared secret.
  - Shared secret used to derive further keys, similar to SSL/IPSec.
  - For confidentiality and authentication in SSH transport layer protocol.
- Secure ciphersuite negotiation.
  - Encryption, MAC, and compression algorithms.
  - Server authentication and key exchange methods.

# SSH-2 Algorithms

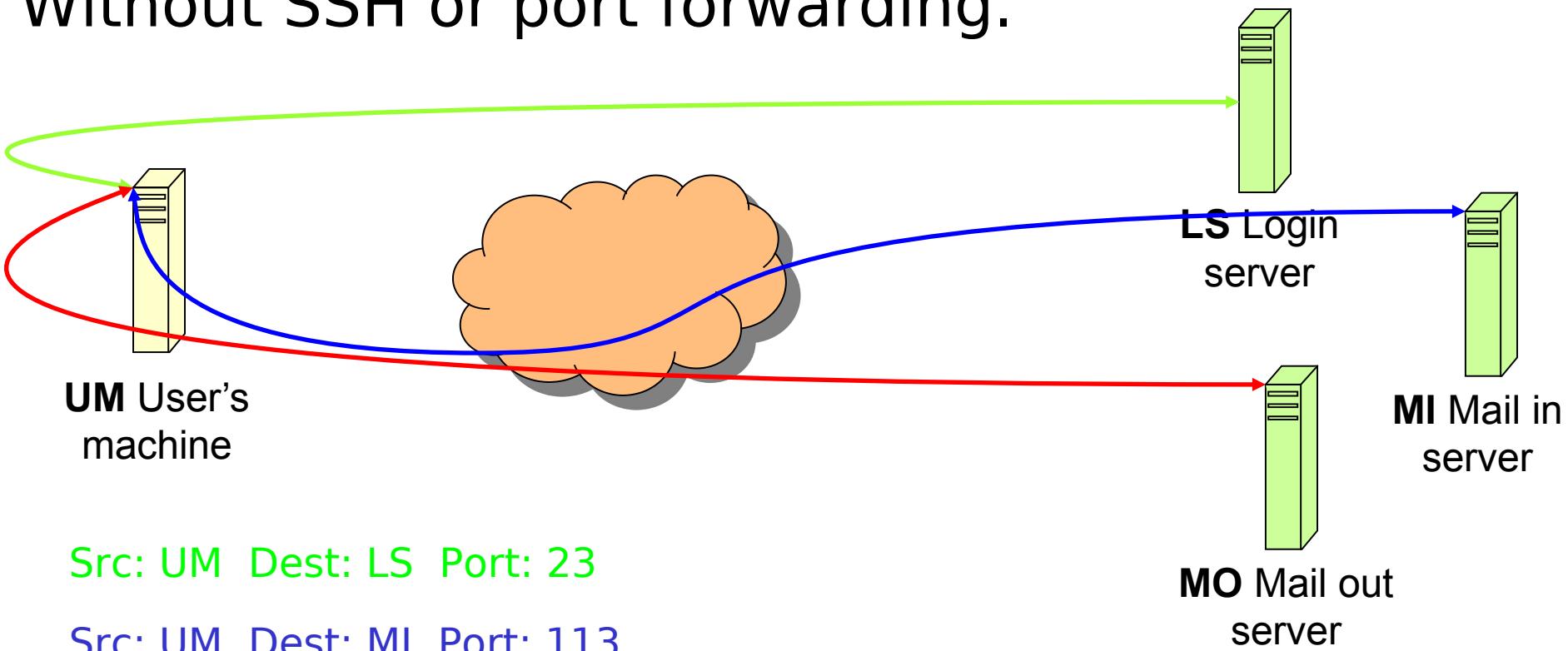
- Key establishment through Diffie-Hellman key exchange.
  - Variety of groups supported.
- Server authentication via RSA or DSS signatures on nonces (and other fields).
- HMAC-SHA1 or HMAC-MD5 for MAC algorithm.
- 3DES, RC4, or AES finalists (Rijndael/Serpent).
- Pseudo-random function for key derivation.
- Small number of ‘official’ algorithms with simple DNS-based naming of ‘private’ methods.

# SSH-1 vs. SSH-2

- Many vulnerabilities have been found in SSH-1 .
  - SSH-1 Insertion attack exploiting weak integrity mechanism (CRC-32) and unprotected packet length field.
  - SSHv1.5 session key retrieval attack (theoretical).
  - Man-in-the-middle attacks (using e.g. dsniff).
  - DoS attacks.
  - Overload server with connection requests.
  - Buffer overflows.
- But SSH-1 widely deployed.
- And SSH-1 supports:
  - Wider range of client authentication methods (.rhosts and Kerberos).
  - Wider range of platforms.

# SSH Port Forwarding

Without SSH or port forwarding.

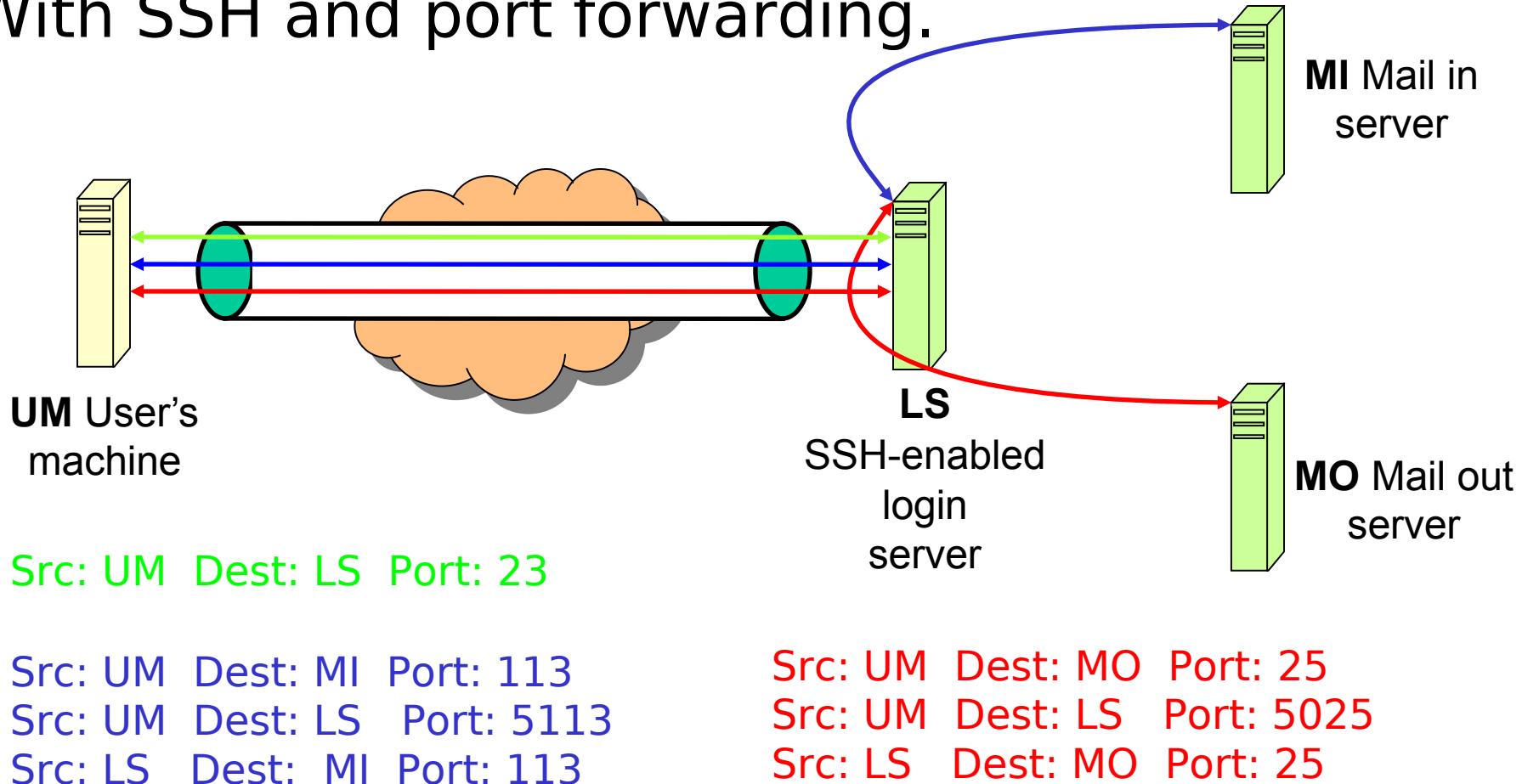


# SSH Port Forwarding

- TCP port number ‘identifies’ application.
- SSH on local machine:
  - Intercepts traffic bound for server.
  - Translates standard TCP port numbers.
  - E.g. port 113 → port 5113.
  - Sends packets to SSH-enabled server through SSH secure channel.
- SSH-enabled server:
  - Receives traffic.
  - Re-translates port numbers.
  - E.g. port 5113 → port 113.
  - Forwards traffic to appropriate server using internal network.

# SSH Port Forwarding

With SSH and port forwarding.



# SSH Applications

- Anonymous ftp for software updates, patches...
  - No client authentication needed, but clients want to be sure of origin and integrity of software.
- Secure ftp.
  - E.g.upload of webpages to webserver using sftp.
  - Server now needs to authenticate clients.
  - Username and password may be sufficient, transmitted over secure SSH transport layer protocol.
- Secure remote administration.
  - SysAdmin (client) sets up terminal on remote machine.
  - SysAdmin password protected by SSH transport layer protocol.
  - SysAdmin commands protected by SSH connection protocol.

# Comparing IPSec, SSL/TLS, SSH

- All three have initial (authenticated) key establishment then key derivation.
  - IKE in IPSec
  - Handshake Protocol in SSL/TLS (can be unauthenticated!)
  - Authentication Protocol in SSH
- All protect ciphersuite negotiation.
- All three use keys established to build a ‘secure channel’.

# Comparing IPSec, SSL/TLS, SSH

- Operate at different network layers.
  - This brings pros and cons for each protocol suite.
  - Naturally support different application types, can all be used to build VPNs.
- All practical, but not simple.
  - Complexity leads to vulnerabilities.
  - Complexity makes configuration and management harder.
  - Complexity can create computational bottlenecks.
  - Complexity necessary to give both flexibility and security.

# Comparing IPSec, SSL/TLS, SSH

Security of all three undermined by:

- Implementation weaknesses.
- Weak server platform security.
  - Worms, malicious code, rootkits,...
- Weak user platform security.
  - Keystroke loggers, malware,...
- Limited deployment of certificates and infrastructure to support them.

Especially client certificates.

- Lack of user awareness and education.
  - Users click-through on certificate warnings.
  - Users fail to check URLs.
  - Users send sensitive account details to bogus websites (“phishing”) in response to official-looking e-mail.

# Types of authentication protocols

# Local Computing

- User sits down in front of the computer
- Responds to the login prompt with a user id and password.
- Machine has a list of all the users and their encrypted passwords
- Password never goes across the network
- Passwords are encrypted with a one-way code
- The crypt algorithm of Unix has been around since mid 70's. Uses a salt to keep identical passwords from having the same encryption. Uses only 8 characters, case sensitive. Uses 25 iterations of DES.
- Typically broken by guessing and verifying guess or snooping the password.

# Remote Access Computing

- User logs in to one or more remote machine(s)
- Each machine has its own copy of userid and password for each user
  - Changing a password on one machine does not affect the other machines
  - Each time a user connects to a different machine, she must login again
- In the standard Unix login or rsh commands, the user's password is sent in clear text over the network or else hosts trust users on the basis of their IP addresses
- Ssh
  - encrypts the password before sending it
  - or uses a user's key pair for establishing her identity

# Single Domain Remote Access Computing

- User gets access to many machines in a single administrative domain.
- He has a single *userid* and *password* for all the machines
- Can login just once to a central trusted server
- Examples
  - Kerberos freeware from MIT Project Athena
  - NIS - Sun software with remote access commands

# Kerberos

# Kerberos

- User - password based authentication based on late-70's Needham -Schroeder algorithms.
- Kerberos Authentication Server aka KDC (Key Distribution Center) shares long-term secret (password) with each authorized user.
- User logs in and established a short term session key with the AS which can be used to establish his identity with other entities, e.g. file system, other hosts or services each of which trusts the authority server.
- The authorization mechanism needs to be integrated with the each function, e.g. file access, login, telnet, ftp, ...
- The central server is a single point of vulnerability to attack and failure.
- Been in use for over 20 years. We are now at version 5.

# Why Kerberos?

- Sending usernames and passwords in the clear text jeopardizes the security of the network.
- Each time a password is sent in clear text, there is a chance for interception.
- Firewalls make a risky assumption: that attackers are coming from the outside. In reality, attacks frequently come from within.



# Kerberos functions

- **Authentication**
- **Integrity** – Is the assurance that the data received is the same as generated.
- **Confidentiality** – is the protection of info from disclosure to those not intended to receive it.
- **Authorization** – is the process by which one determines whether a principal is allowed to perform an operation. Authorization is done usually after principal has been authenticated or based on authenticated statements by others.

# Kerberos Credentials

- **Ticket**

- Allows user to use a service (actually authenticate to it)
- Used to securely pass the identity of the user to which the ticket is issued between the KDC and the application server
- $K_b\{\text{"alice"}, K_{ab}, \text{lifetime}\}$

- **Authenticator**

- Proves that the user presenting the ticket is the user to which the ticket was issued
- Proof that user knows the session key
- Prevents ticket theft from being useful
- Prevents replay attacks (timestamp encrypted with the session key):  $K_{ab}\{\text{timestamp}\}$ , in combination with a replay cache on the server

# Kerberos with TGS

- **Ticket Granting Service (TGS):**
  - A Kerberos authenticated service, that allows user to obtain tickets for other services
  - Co-located at the KDC
- **Ticket Granting Ticket (TGT):**
  - Ticket used to access the TGS and obtain service tickets
- **Limited-lifetime session key: TGS sessionkey**
  - Shared by user and the TGS
- TGT and TGS session-key cached on the user's workstation

# Cryptography Approach

- Private Key: Each party uses the same secret key to encode and decode messages.
- Uses a trusted third party which can vouch for the identity of both parties in a transaction. Security of third party is imperative.

# Kerberos cryptography

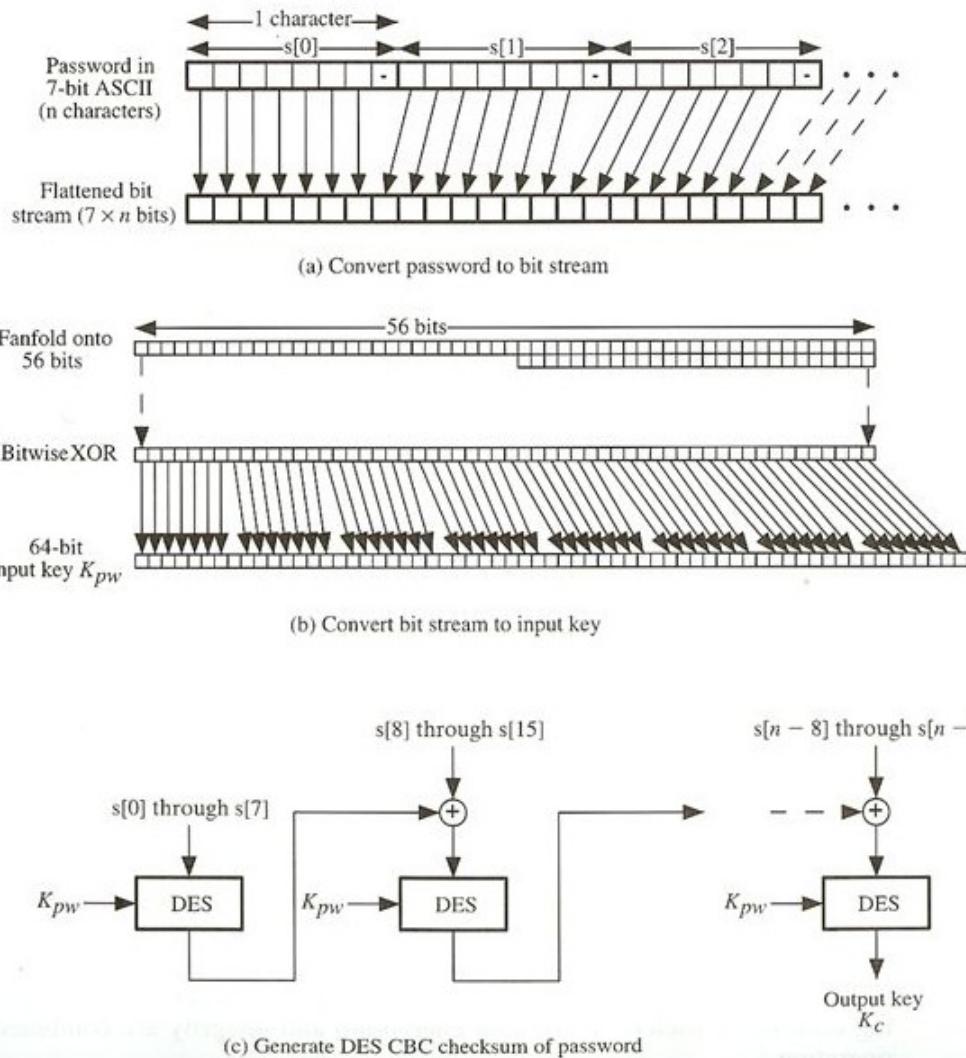


Fig. 4.6 Generation of Encryption Key from Password

# Kerberos cryptography

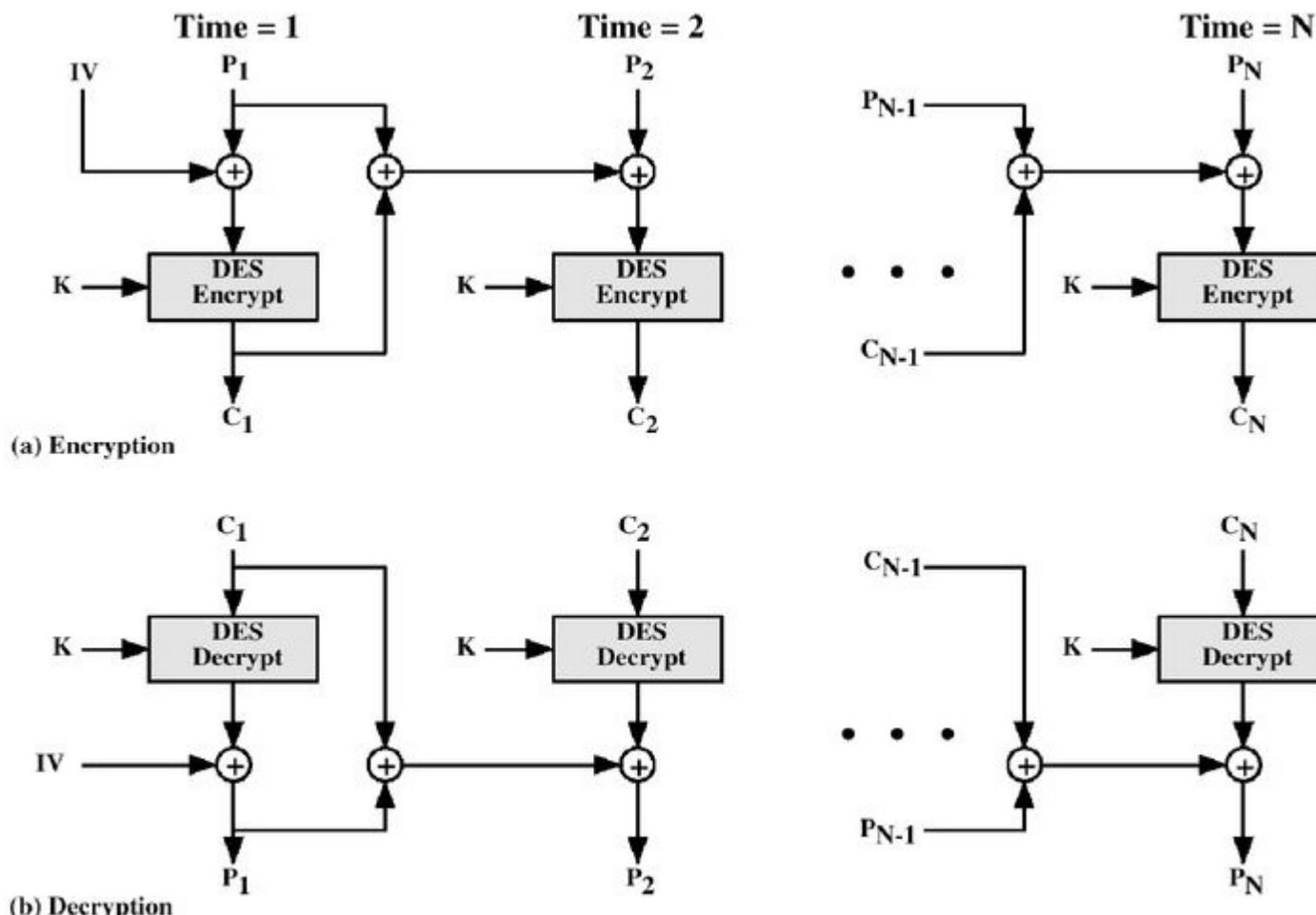


Figure 4.7 Propagating Cipher Block Chaining (PCBC) Mode

## TGS Benefits

- Single Sign-on (SSO) capability
- Limits exposure of user's password
  - Alice's workstation can forget the password immediately after using it in the early stages of the protocol
  - Less data encrypted with the user's secret key travels over the network, limiting attacker's access to data that could be used in an offline dictionary attack

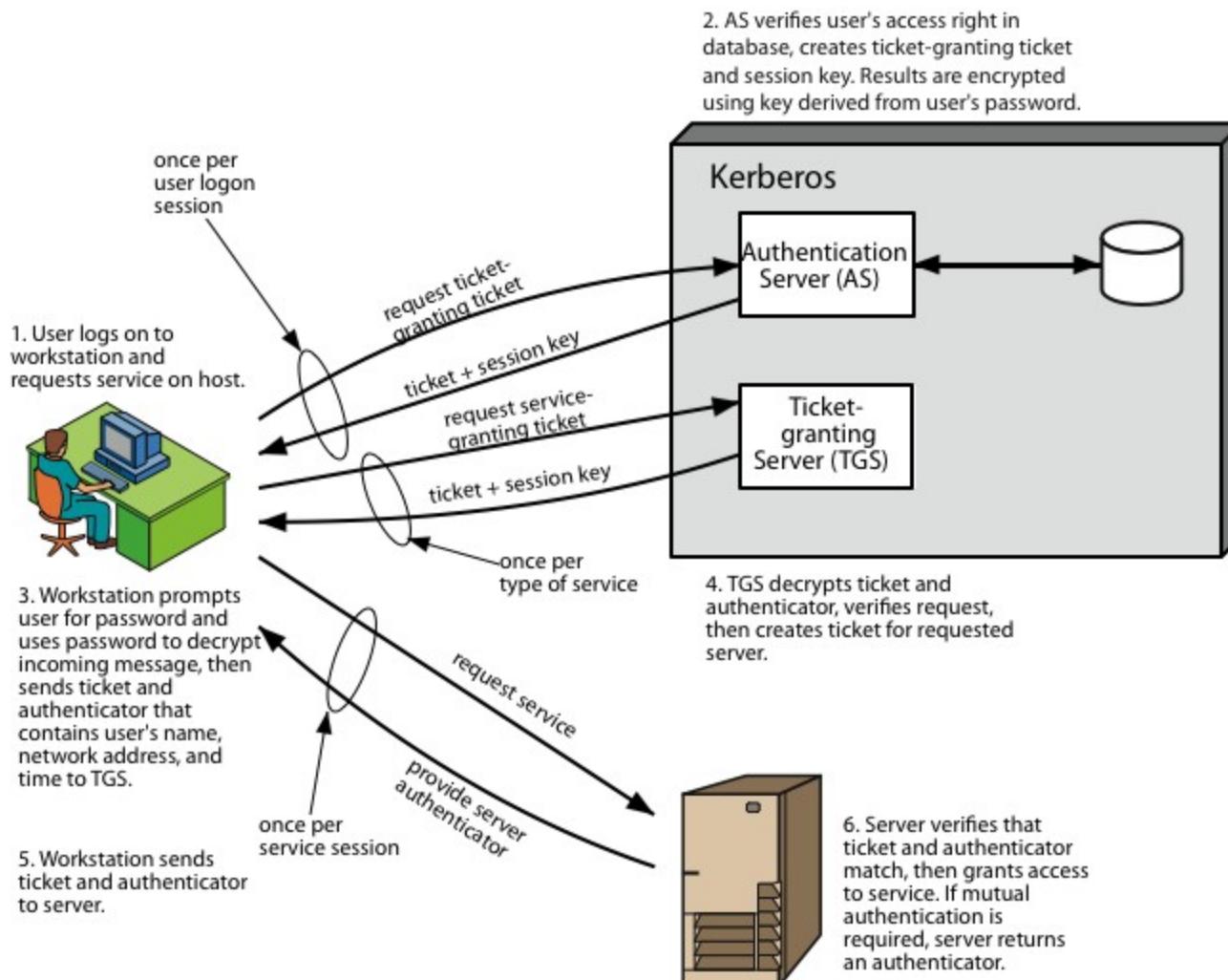
# How does Kerberos work?

- Instead of client sending password to application server:
  - Request **Ticket** from authentication server
  - Ticket and encrypted request sent to application server
- How to request tickets without repeatedly sending credentials?
  - **Ticket granting ticket (TGT)**

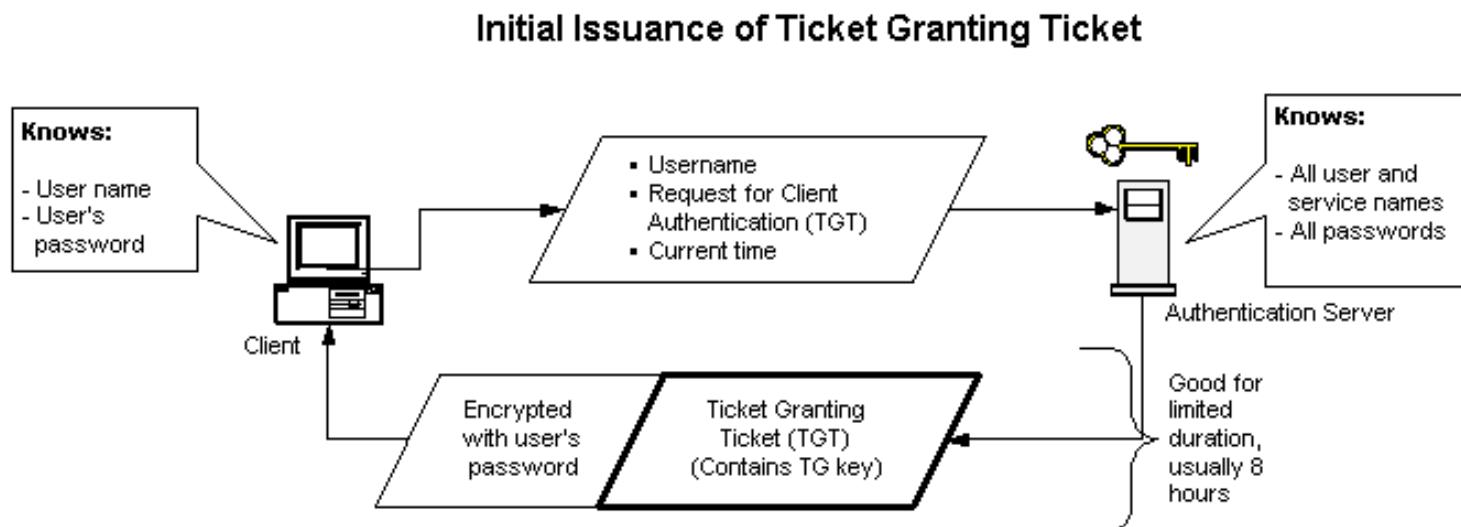
# How does Kerberos work?

- User presents a ticket that is issued to it by a Kerberos Authentication Server(AS).
- If the ticket is valid, service is granted.
- The tickets must be unequivocally linked to the user
- Ticket demonstrates that the bearer knows something that only its intended user would know.
- Ticket must obviously be safeguarded against all attacks.

# How does Kerberos work?

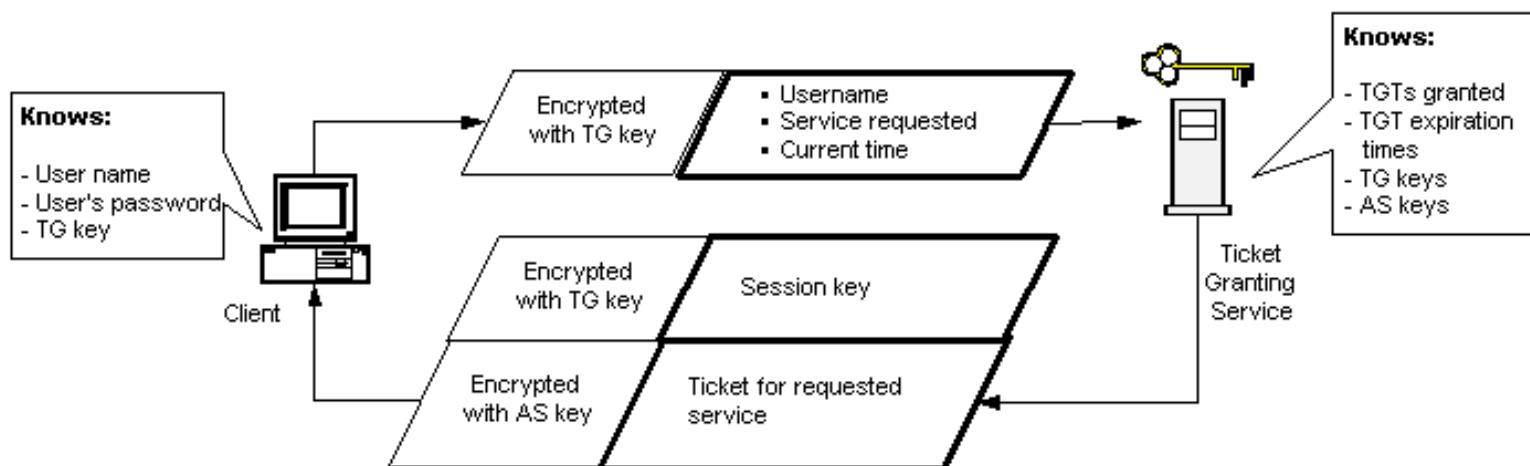


# How does Kerberos work?: Ticket Granting Tickets



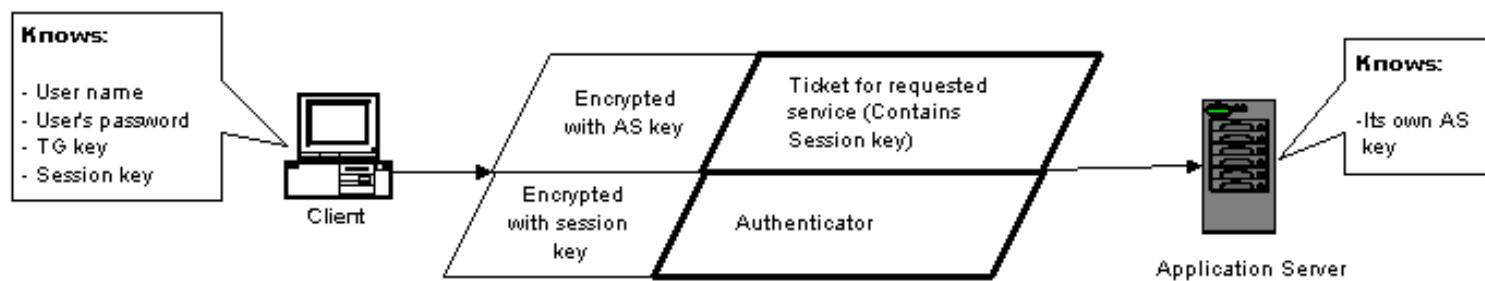
# How does Kerberos Work?: The Ticket Granting Service

## Susequent Requests for Services from the Ticket Granting Service



# How does Kerberos work?: The Application Server

Communication between the Client and the Application Server



# Weaknesses and Solutions

If TGT stolen, can be used to access Network services.

Only a problem until ticket expires in a few hours.

Subject to dictionary attack.

Timestamps require hacker to guess in 5 minutes.

Very bad if Authentication Server compromised.

Physical protection for the server.

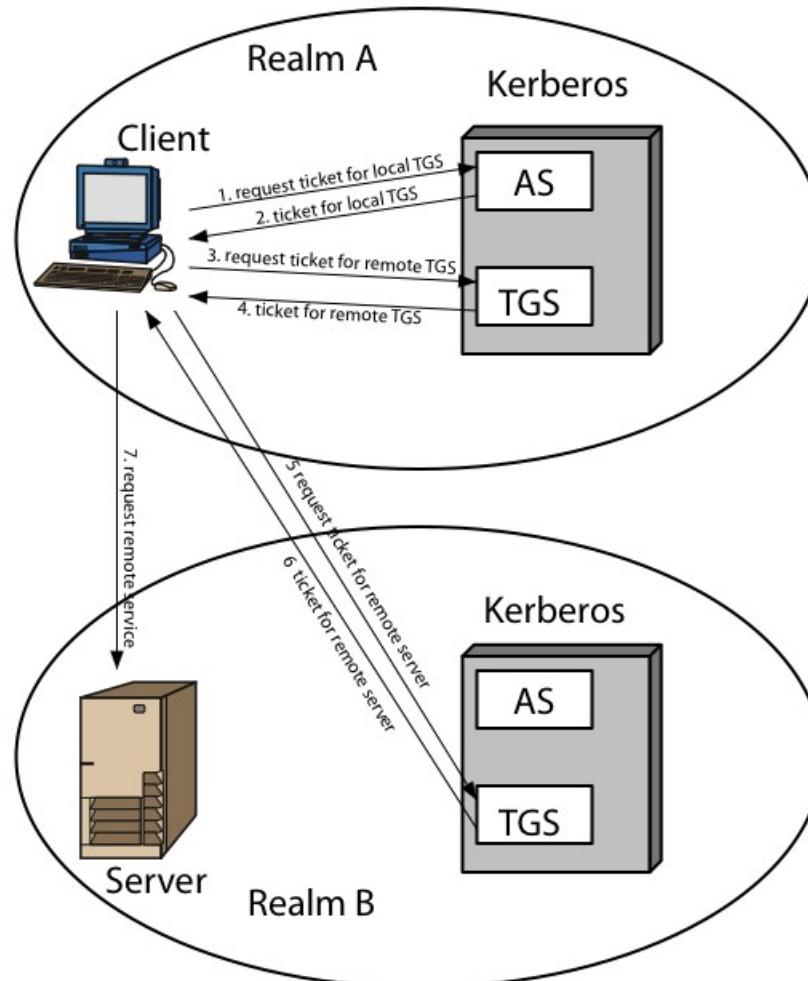
# The Competition: SSL

SSL	Kerberos
Uses public key encryption	Uses private key encryption
Is certificate based (asynchronous)	Relies on a trusted third party (synchronous)
Ideal for the WWW	Ideal for networked environments
Key revocation requires Revocation Server to keep track of bad certificates	Key revocation can be accomplished by disabling a user at the Authentication Server
Certificates sit on a users hard drive (even if they are encrypted) where they are subject to being cracked.	Passwords reside in users' minds where they are usually not subject to secret attack.
Uses patented material, so the service is not free. Netscape has a profit motive in wide acceptance of the standard.	Kerberos has always been open source and freely available.

# Kerberos Realms

- Cross-realm trust is possible
- Just need to share a secret key between the KDCs for the two realms...
- Once accomplished, a user in realm A can get a ticket for a service in realm B

# Kerberos Realms



# Advantages of Kerberos (1)

- Passwords aren't exposed to eavesdropping
- Password is only typed to the local workstation
  - It never travels over the network
  - It is never transmitted to a remote server
- Password guessing more difficult
- Single Sign-on
  - More convenient: only one password, entered once
  - Users may be less likely to store passwords
- Stolen tickets hard to reuse
  - Need authenticator as well, which can't be reused
- Much easier to effectively secure a small set of limited access machines (the KDC's)
- Easier to recover from host compromises
- Centralized user account administration

## Advantages of Kerberos (2)

- More efficient authentication to servers.
  - Server can authenticate the client by examining credentials. Clients can obtain credentials for a particular server once and reuse them throughout a network logon session.
- Mutual authentication.
  - Parties at both ends of a network connection can know that the party on the other end is who it claims to be.
- Delegated authentication.
  - Kerberos protocol has a proxy mechanism that allows a service to impersonate its client when connecting to other services.

# Drawbacks of Kerberos

- Single point of failure: It requires continuous availability of a central server. When the Kerberos server is down, no one can log in.
- Kerberos has strict time requirements, which means the clocks of the involved hosts must be synchronized within configured limits.
- The administration protocol is not standardized and differs between server implementations.
- Since all authentication is controlled by a centralized KDC, compromise of this authentication infrastructure will allow an attacker to impersonate any user.
- Each network service which requires a different host name will need its own set of Kerberos keys. This complicates virtual hosting and clusters.
- KDC could be a performance bottleneck.
  - Everyone needs to communicate with it frequently.
  - Not a practical concern these days
  - Having multiple KDC's alleviates the problem

# Conclusions (1)

- Authentication is critical for the security of computer systems. Without knowledge of the identity of a principal requesting an operation, it's difficult to decide whether the operation should be allowed.
- Traditional authentication methods are not suitable for use in computer networks where attackers monitor network traffic to intercept passwords.
- The use of strong authentication methods that do not disclose passwords is imperative. The Kerberos authentication system is well suited for authentication of users in such environments.

## Conclusions (2)

- Only the KDC needs to know the user's password (used to generate the shared secret)
- You can have multiple KDCs for redundancy, but they all need to have a copy of the username/password database
- Only the TGS needs to know the secret keys for the servers You can split KDC from TGS, but it is common for those two services to reside on the same physical machine

## Conclusions (3)

- “Time” is very important in Kerberos
- All participants in the realm need accurate clocks
- Timestamps are used in authenticators to detect replay; if a host can be fooled about the current time, old authenticators could be replayed
- Tickets tend to have lifetimes on the order of hours, and replays are possible during the lifetime of the ticket

# PKI

## Public Key Infrastructure

# Digital Signatures

- Combines a hash with a digital signature algorithm
- To sign
  - hash the data
  - Encrypt the hash with the sender's private key
  - send data signer's name and signature
- To verify
  - hash the data
  - find the sender's public key
  - decrypt the signature with the sender's public key
  - the result of which should match the hash

# Digital certificates

A certificate acts as a trusted “third party” which allows the authentication of an entity even when it is the first contact with that entity.

- A certificate is issued by a Certificate Authority(CA)
- Digital certificates are created according to the X.509 standard.

I hereby certify that the public key

19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A  
belongs to

Robert John Smith

12345 University Avenue

Berkeley, CA 94702

Birthday: July 4, 1958

Email: bob@superduper.net.com

SHA-1 hash of the above certificate signed with the CA's private key

# Digital certificates

A certificate is not secret. It is signed by the CA which issued it.

Checking a certificate:

- The SHA-1 hash of the certificate is computed
- The signature of the certificate is decrypted using the public key of the CA.
- The two results are compared.

I hereby certify that the public key

19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A

belongs to

Robert John Smith

12345 University Avenue

Berkeley, CA 94702

Birthday: July 4, 1958

Email: bob@superduper.net.com

SHA-1 hash of the above certificate signed with the CA's private key

# Public key certificates

A **public key certificate** is a set of data that binds an identity to a particular public key value. The four core pieces of information that are contained in a public key certificate are as follows:

- **Name of owner**
  - could be a person, device, or even role.
  - Should uniquely identify the owner within the environment in which the public key will be employed.
- **Public key value**
- **Validity time period**
  - identifies date and time from which the public key is valid, and more importantly the date and time of its expiry.
- **Signature**
  - Creator of certificate digitally signs all data that forms the public key certificate. This binds the data and acts as a guarantee that the creator of the certificate believes the data is correct.

# X509 v3 public key certificates

<b>Version</b>	This specifies the X.509 version being used (in this case v3).
<b>Serial Number</b>	A unique identifier for the certificate.
<b>Signature</b>	The digital signature algorithm used to sign the certificate.
<b>Issuer</b>	The name of the creator of the digital certificate.
<b>Validity</b>	The dates and times between which the digital certificate is valid.
<b>Subject</b>	The name of the owner of the digital certificate.
<b>Subject Public Key Info</b>	The actual public key and the identifier of the public key algorithm associated with it.
<b>Issuer Unique ID</b>	An optional identifier for the creator of the digital certificate.
<b>Subject Unique ID</b>	An optional identifier for the owner of the digital certificate.
<b>Extensions</b>	<p>A range of optional fields that include:</p> <ul style="list-style-type: none"> <li>• a key identifier (in case owner owns more than one public key)</li> <li>• key usage information that specifies valid uses of key</li> <li>• the location of revocation information</li> <li>• identifier of the certificate policy</li> <li>• alternative names for the owner</li> </ul>

# Certificate authorities

It should be clear that the “creator” of a public key certificate plays an extremely important role.

A creator of a public key certificate is normally referred to as a **Certificate Authority** (or **CA**).

**The CA takes responsibility for ensuring that the information on a certificate is correct. The CA creates (or issues) the public key certificate to the owner.**

**Whenever anyone has need of the owner's public key they request a copy of the public key certificate. The certificate might be made available on a central server, or the owner or even the CA might send the certificate to whoever requires it.**

**The recipient of the public key certificate checks that the certificate is in order, and if they are happy with it then they are free to use the public key contained in the certificate.**

# Trusting a digital certificate

There are three things that the recipient “needs to be able to do” in order to be satisfied that the public key certificate is in order:

**The recipient needs to be able to trust (directly or indirectly) the CA to have done their job correctly and to have gone through some process to verify all the fields of the certificate.**

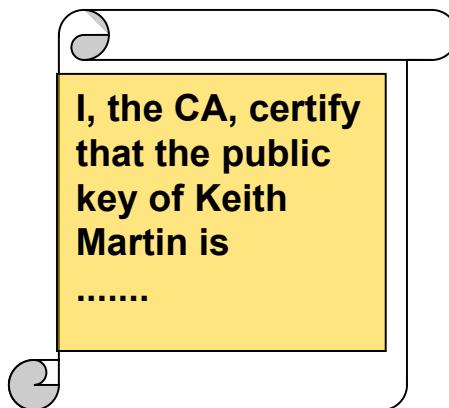
**The recipient needs to have access to the public verification key of the CA in order to verify the CA’s digital signature on the certificate.**

**The recipient needs to check all the fields in the certificate. In particular they must check that the certificate is valid, it applies to the correct owner and that the other fields are all satisfactory.**

For each of the above three recipient checks, what are the precise implications of them not being done?

# Meaning of certificates

By digitally signing the information in the public key certificate, the Certification Authority is effectively making the statement:



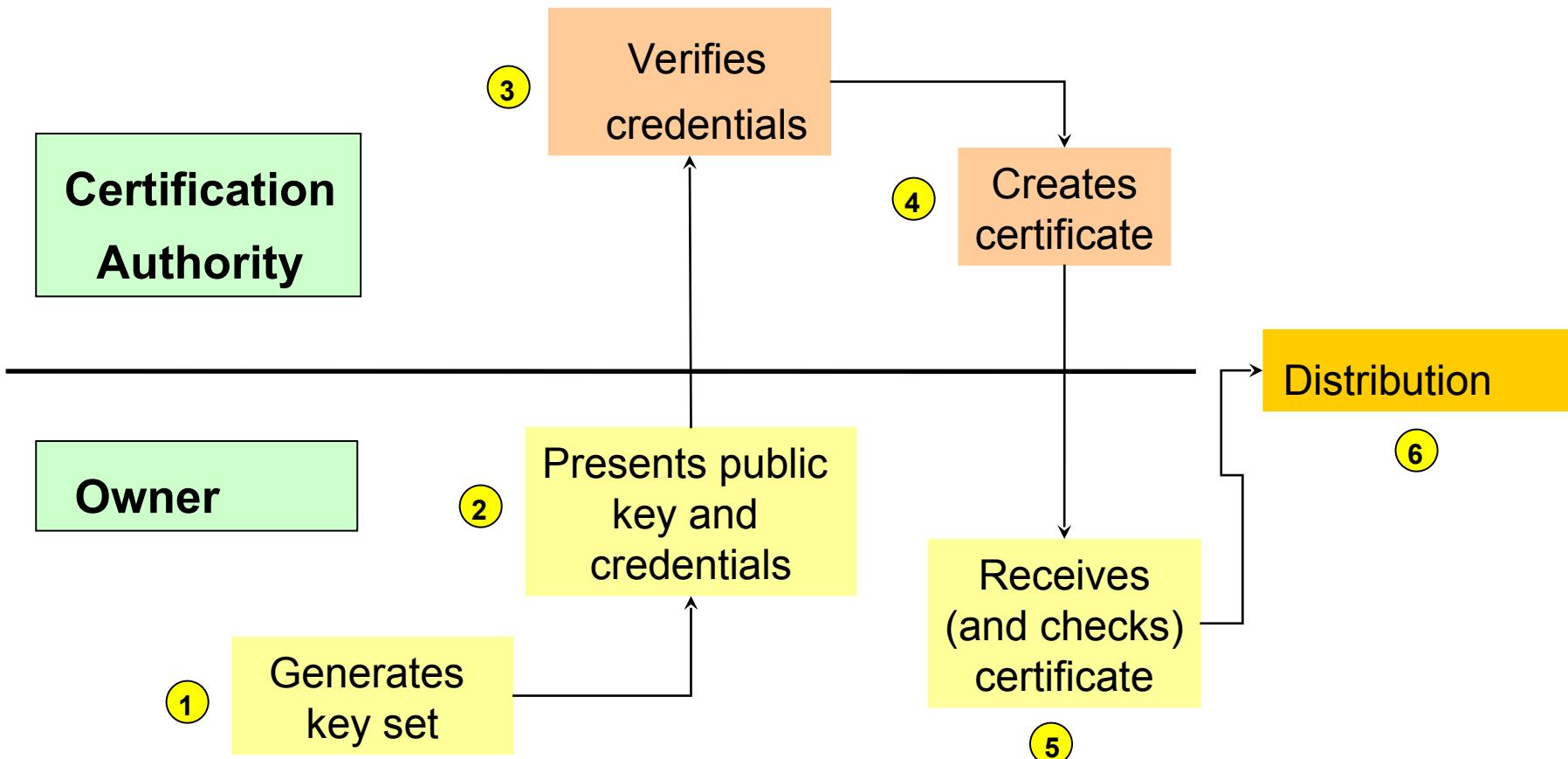
# Certificate Authority

- A trusted third party - must be a secure server
- Signs and publishes X.509 Identity certificates
- Revokes certificates and publishes a Certification Revocation List (CRL)
- Many vendors
  - OpenSSL - open source, very simple
  - Netscape - free for limited number of certificates
  - Entrust - Can be run by enterprise or by Entrust
  - Verisign - Run by Verisign under contract to enterprise
  - RSA Security - Keon servers

# Key/Certificate Life Cycle

- Initialization
  - Registration
  - Key-pair generation (where?)
  - Certificate creation and dissemination
  - Key backup
- Issued
  - Certificate retrieval
  - Certificate validation
- Cancellation
  - Expiration
  - Revocation
  - History and archive

# Example certificate issuing process



# Generating the public key pair

## Pros

The owner is placed in full control of their own key material.

It may be easier and more secure to manage the generation of key material centrally.

The certification process might appear more seamless to the owner.

## Owner

## Issuer (CA)

## Cons

The owner might not have the capability or skill to perform this operation in a secure fashion.

The owner needs to prove to the CA that they actually know the private key before the certificate can be issued.

The owner must trust the CA to securely deliver the private key to the owner and to dispose of it afterwards.

# Registration

The stage of the certification process at which the owner presents their credentials to the CA for checking is arguably the most vital stage in the entire certification process.

In many application environments a separate entity known as a **Registration Authority (RA)** performs this operation.

There are two arguments for keeping the roles of CA and RA at least slightly separate:

Most of the functionality of a CA can be essentially performed by a computer, whereas for many applications the role of the RA requires human intervention.

Checking the credentials of a certificate applicant is often the most complex part of the certification process. There is thus a strong argument for distributing the registration activities across a number of “local” RAs.

# Levels of certificate

It is worth noting that many CAs issue different types of certificate (sometimes referred to as **levels** of certificate) depending upon the thoroughness of the process used to identify the owner.

These levels often correspond directly to the credentials by which the owner was identified when the certificate was registered.

Certificates of different levels may then be used in different types of application. The liability associated with a certificate is likely to be different for different levels of certificate.

# Proof of possession

However registration is done, there is one very important check that must be performed before proceeding with the issuing of a public key certificate – that the owner actually knows the private key corresponding to the public key in the certificate.

If the CA does the key generation then this problem does not arise, but if the owner generates the key pair then this check is essential.

This process is referred to as demonstrating **Proof of Possession**.

# Certificate distribution

## • **Pushing**

- **the owner of the certificate automatically provides the certificate when it is required**
- **the problem with pushing is that the receiver of the certificate needs to check that the certificate that they have just received is still valid.**

## • **Pulling**

- **users must request copies of certificates when they need them.**
- **the problem with pulling is that this requires the relevant CAs to be online to distribute the certificates when required to do so.**
- **an advantage is that the receiver is more likely to get the latest valid certificate, although it may still be prudent that the receiver performs checks to ensure that the certificate has not been revoked.**

# Certifying the certifiers

Someone, somewhere, must generate the CA's public key pair, and someone, somewhere, must certify this public key. However it is done, it must be done securely. If someone gets hold of the private key of the CA then they can generate certificates themselves, and the whole system falls apart.

# Revocation

We must consider how to handle certificates that need to be “withdrawn” before their expiry date. This process is often referred to as **certificate revocation**.

- **Certificate Revocation List (or CRLs)**

A lists of certificates that have been revoked.

CRLs need to be maintained carefully, with clear indications of how often they are updated.

CRLs need to be signed by the CA and be made available to users as easily as possible.

- **Online Certificate Status Protocol (OCSP)**

An online database containing the status of certificates issued by the CA .

# The End

# Security Protocols

Lecture  
6

# Advanced Access Control



# RADIUS

# The RADIUS Protocol

- The Remote Authentication Dial-In User Service (RADIUS) protocol was developed by Livingston Enterprises, Inc., as an access server authentication and accounting protocol.

*Current standard for remote authentication*

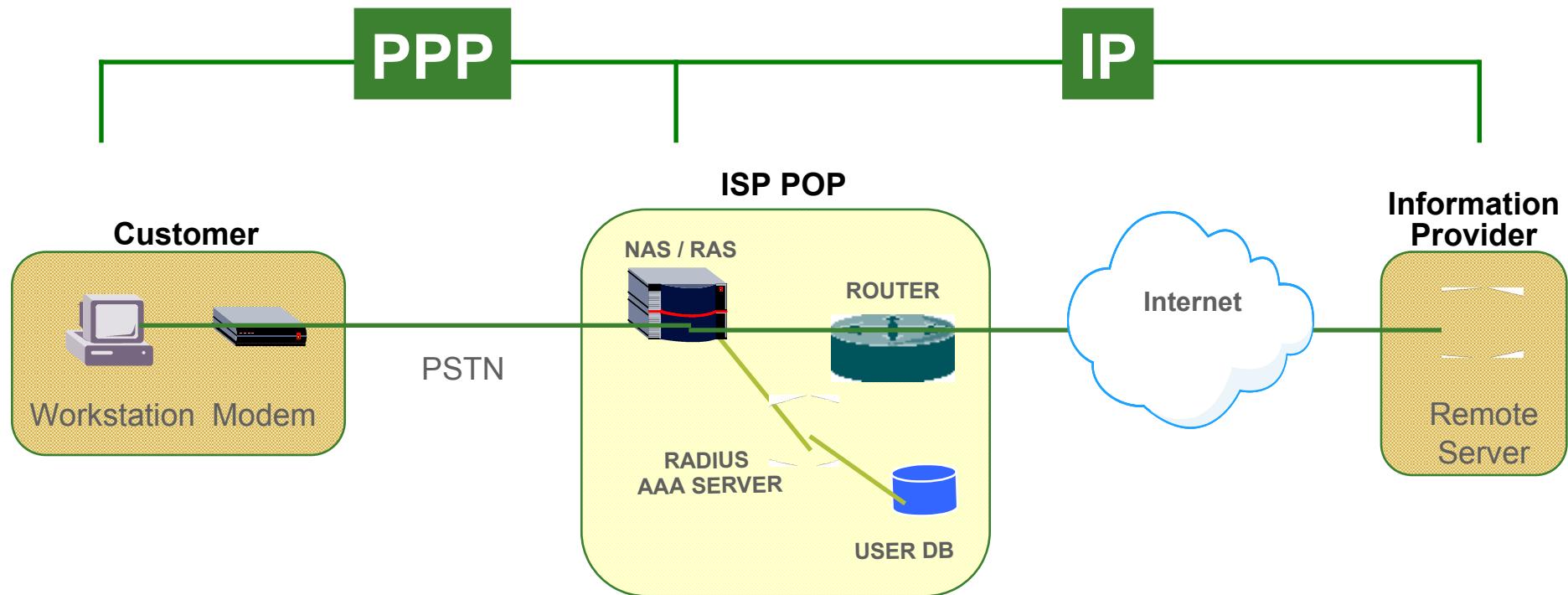
*Current versions of RADIUS protocol:*

*RFC 2865 (RADIUS)*

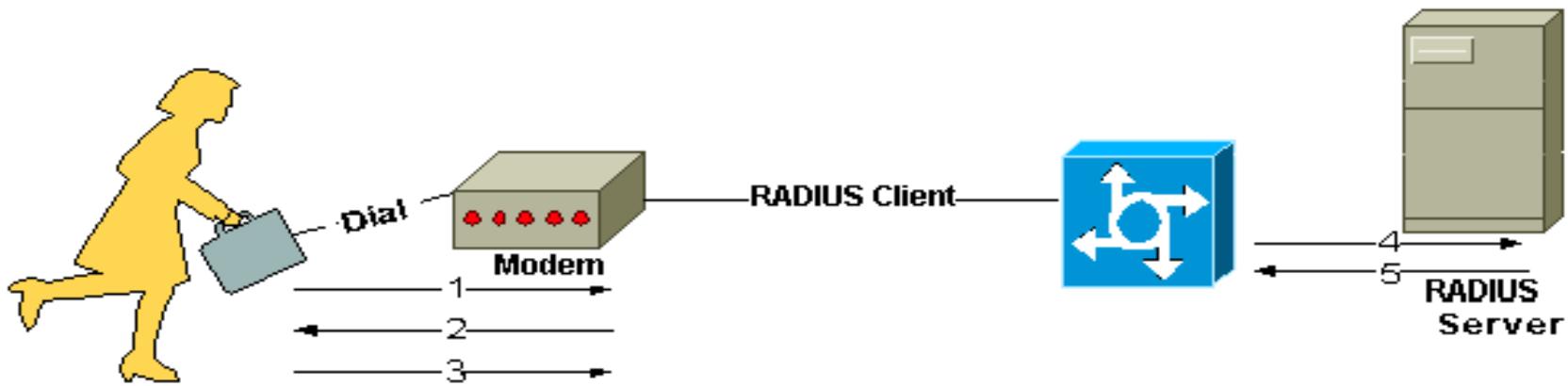
*RFC 2866 (RADIUS Accounting)*

- AAA protocol (Authentication, Authorization and Accounting)
- Supports applications such as
  - Network access
  - IP mobility
- Used in embedded network devices such as modems servers, routers, switches
- Works in both local and roaming situations

# Logical System View



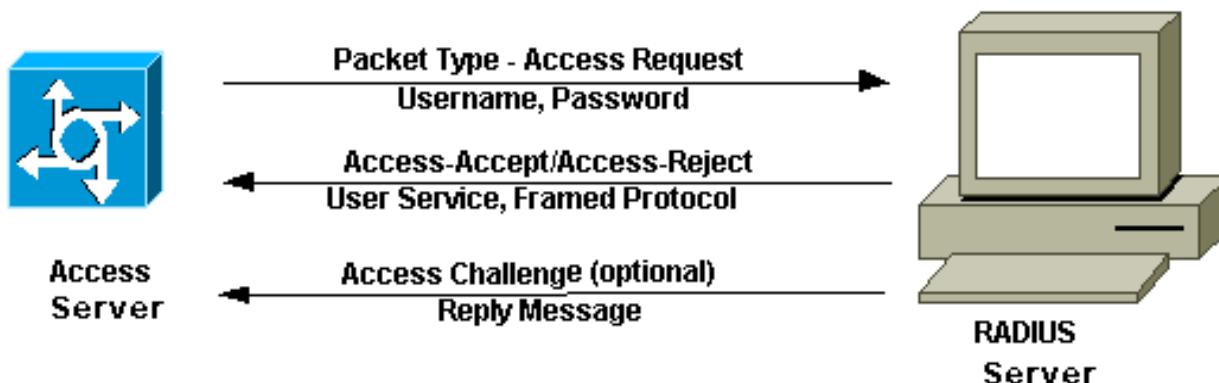
# The RADIUS Protocol



- 1) User initiates PPP authentication to the NAS.
- 2) NAS prompts for username and password (if Password Authentication Protocol [PAP]) or challenge (if Challenge Handshake Authentication Protocol [CHAP]).
- 3) User replies.
- 4) RADIUS client sends username and encrypted password to the RADIUS server.
- 5) RADIUS server responds with Accept, Reject, or Challenge.
- 6) The RADIUS client acts upon services and services parameters bundled with Accept or Reject.

# The RADIUS Protocol

- In RADIUS, authentication and authorization are coupled together.
- If the username is found and the password is correct, the RADIUS server returns an Access-Accept response, including a list of attribute-value pairs that describe the parameters to be used for this session. Typical parameters include:
  - service type (shell or framed),
  - protocol type,
  - IP address to assign the user (static or dynamic),
  - access list to apply, or a static route to install in the NAS routing table.



# The RADIUS Protocol

## Protocol messages:

- **Access-Request** : authentication and authorization for a connection attempt by a RADIUS client

Possible responses from the server to the client:

- **Access-Accept** : connection attempt is authenticated and authorized. Provides specific configuration information necessary to begin the delivery of service to the user.
- **Access-Reject** : issued by the server when unacceptable attributes are received.
- **Access-Challenge** : demands challenge response and causes access deny for subscribers.

# The RADIUS Protocol

- Communication between a network access server (NAS) and a RADIUS server is based on the User Datagram Protocol (UDP).
- The RADIUS protocol is considered a connectionless service. Issues related to server availability, retransmission, and timeouts are handled by the RADIUS-enabled devices rather than the transmission protocol.
- Network Security:
  - transactions authenticated using shared secret key, manually distributed, between client/server
  - any user passwords, hidden using MD5 (RFC1321) + other
  - End-to-end security (for non proxy RADIUS), but cannot be guaranteed

# The RADIUS Protocol

- RADIUS is a client/server protocol.
  - The RADIUS client is typically a NAS and the RADIUS server is usually a daemon process running on a UNIX or Windows NT machine.
  - The client passes user information to designated RADIUS servers and acts on the response that is returned.
  - RADIUS servers receive user connection requests, authenticate the user, and then return the configuration information necessary for the client to deliver service to the user.
  - A RADIUS server can act as a proxy client to other RADIUS servers or other kinds of authentication servers

# RADIUS Details

- RADIUS uses UDP instead of TCP as a transport protocol
- Some of the reasons for using UDP:
  - User can wait for only few seconds. It can't wait for several minutes
  - No special handling for rebooting or offline clients and servers
  - Stateless protocol
  - Easy to implement multi-threaded server to service multiple client requests

# The RADIUS Protocol

- **Advantages**

- If request to primary authentication server fails, secondary server must be queried:
  - copy of request must be kept above transport layer
  - retransmission timers still required, above transport layer
- Stateless nature of RADIUS protocol within communication network simplifies use of UDP:
  - transport connection between client/server remains even if network failures are occurring

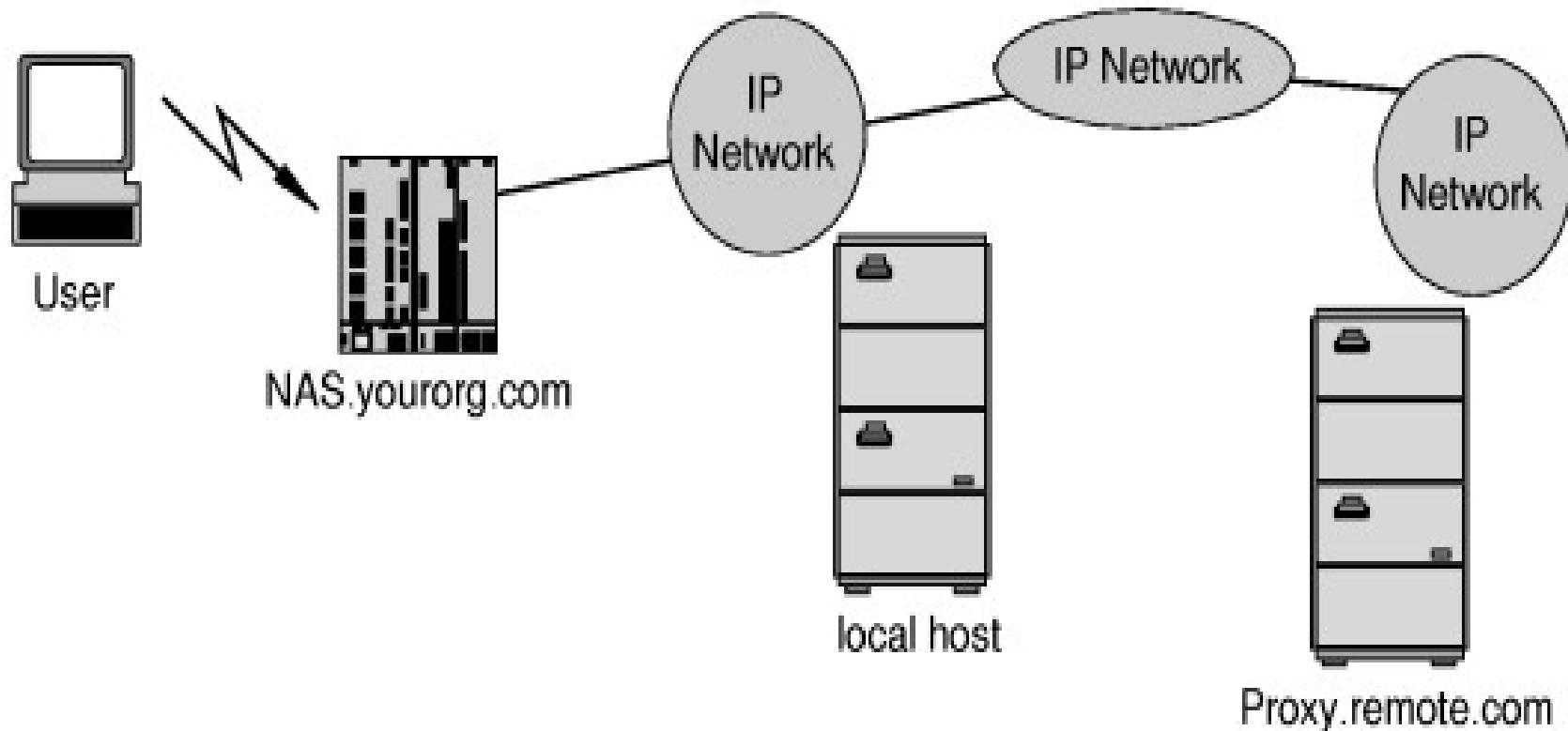
- **Disadvantages**

- Transport is not reliable (layer above transport has to take care of packet losses)
- TCP adapt to network congestion, while UDP does not

# RADIUS Proxying

- The proxy feature forwards authentication (and accounting) to another server
- Used for
  - Carriers
  - Roaming users
  - Applications where different organizations use shared resources

# Proxy Setup

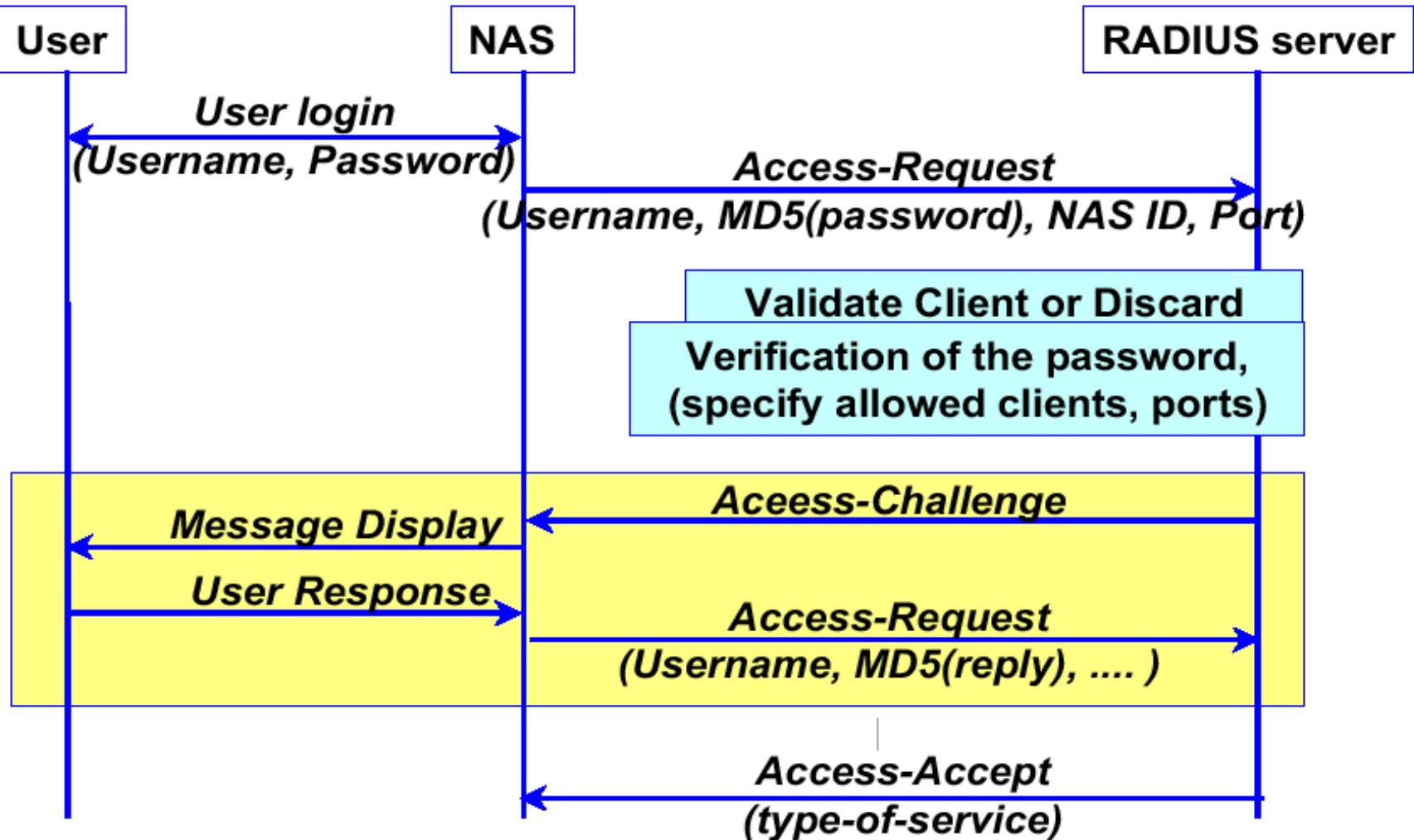


# RADIUS Proxy Servers

- Client sends access request to forwarding server
- Gets forwarded to remote server
- Remote server sends access-accept
- Forwarding server sends the access accept to the client

- When a client is configured to use RADIUS, any user of the client presents authentication information to the client.
- Once the client has obtained such information, it may choose to authenticate using RADIUS. To do so, the client creates an "**Access-Request**" containing such Attributes as the user's name, the user's password, the ID of the client and the Port ID which the user is accessing.
- The Access-Request is submitted to the RADIUS server via the network.
- Once the RADIUS server receives the request, it validates the sending client. A request from a client for which the RADIUS server does not have a shared secret MUST be silently discarded.

# Authentication Flow



# Authentication Flow

If any condition is not met, the RADIUS server sends an "**Access-Reject**" response indicating that this user request is invalid.

If all conditions are met and the RADIUS server wishes to issue a challenge to which the user must respond, the RADIUS server sends an "**Access-Challenge**" response.

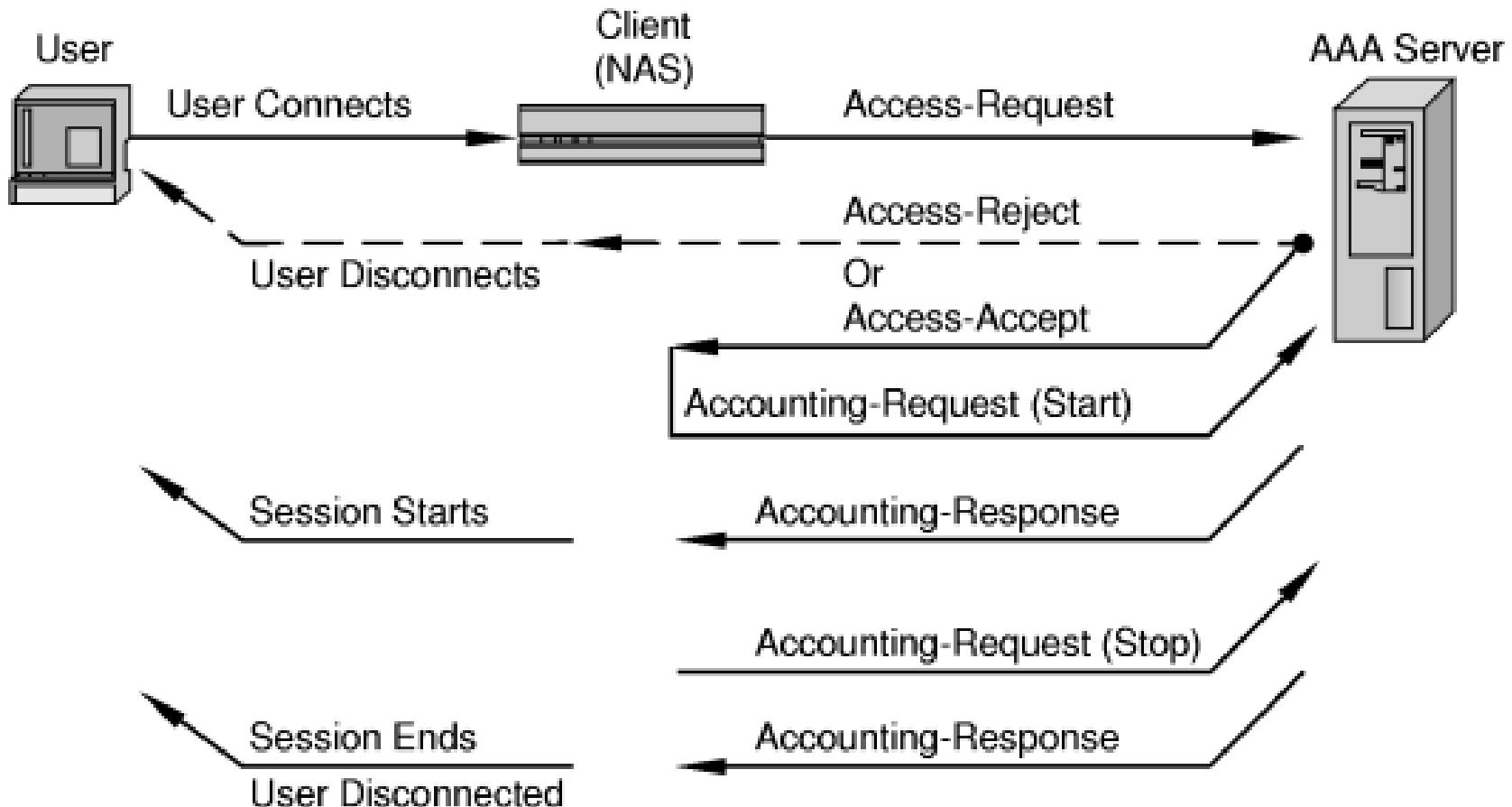
If the client receives an Access-Challenge and supports challenge/response it MAY display the text message, if any, to the user, and then prompt the user for a response. The client then re-submits its original **Access-Request** with a new request ID .

If all conditions are met, the list of configuration values for the user are placed into an "**Access-Accept**" response.

# Accounting messages

- ***Accounting-Request*** : sent by the RADIUS client to specify accounting information for the connection accepted
- ***Accounting-Response*** : acknowledges the successful receipt and processing of Accounting-Request message

# Client-server Transaction



# Accounting Process

- At the start of service delivery it will generate an **Accounting Start packet** describing the type of service being delivered and the user it is being delivered to, and will send that to the RADIUS Accounting server, which will send back an acknowledgement that the packet has been received.
- The Accounting-Request (whether for Start or Stop) is submitted to the RADIUS accounting server via the network. It is recommended that the client continue attempting to send the Accounting-Request packet until it receives an acknowledgement, using some form of backoff.

# Packet Details

- **Code** (8 bits) indicates the type of RADIUS packet
- The table shows the codes assigned to packet types
- 255 is reserved for future use
- **Packet with invalid code is discarded**

Value	Description
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request
5	Accounting-Response
11	Access-Challenge
12	Status-Server (experimental)
13	Status-Client (experimental)

# Packet Details

- ***Identifier*** (8 bits) helps in matching requests and response
- Server can use the identifier to detect duplicate requests from the same client IP address
- Identifiers must be reused frequently

# Packet Details

- **Length** (16 bits) indicates the entire length of the RADIUS packet
- If packet received was shorter than **Length** , then it is dropped
- The extra bits are ignored, if packet is longer than **Length**
- Minimum length is 20 bits and maximum is 4096 bits

# Packet Details

**Authenticator** (16 bytes) used to authenticate the reply from the RADIUS server

- Different for both access and accounting requests and responses

## **Request Authenticator:**

- The value should be unique and unpredictable random number over the entire lifetime of the secret key
- Secret key followed by Request Authenticator is put through MD5 hash, then XORed with user password
- Result is placed in the password attribute

## **Response Authenticator:**

- The values of authentication fields in all access responses indicate Response Authenticator

# Packet Details

## **Response Authenticator:**

- MD5 hash over concatenated fields

*hash(Code | ID | Length | Request Authenticator | Attributes | Secret key )*

## **Request Authenticator:**

- MD5 hash over concatenated fields

*hash(Code | ID | Length | Request Authenticator | Response Attributes | Shared secret )*

# Vulnerability

- RADIUS hiding method ( MD5 hash and stream cipher) may not be adequate.
- Client Access-Request message is not authenticated.
- Request Authenticators may be poorly implemented.
- Administrators may choose the RADIUS shared secrets poorly.
- Multiple clients sharing the same secret make the key easier to discover.

# Conclusion

- RADIUS is Commonly used in embedded systems (routers, switches, etc), which cannot handle large numbers of users with distinct authentication information.
- Facilitates centralized user administration (useful for ISPs)
- Other alternatives have less security.
- Widely implemented by hardware vendors.

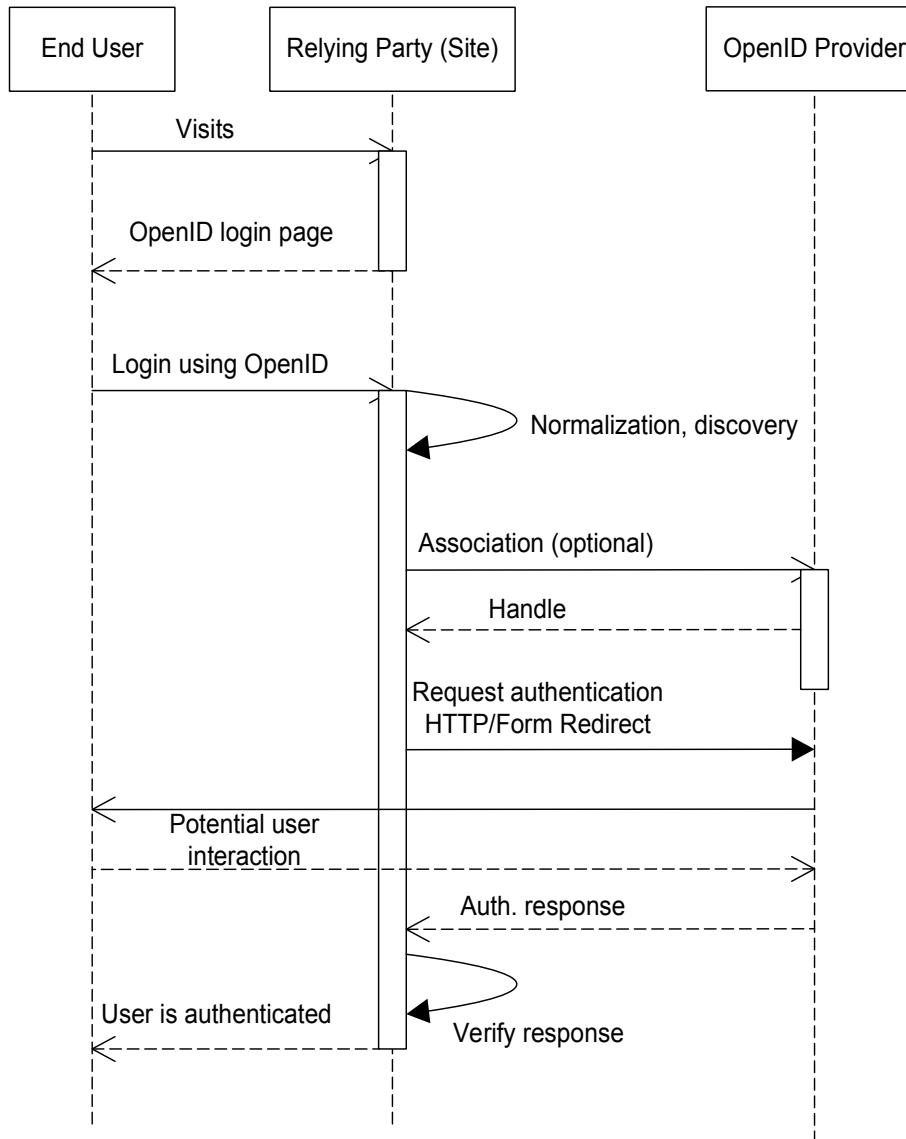
# OpenID

# Motivation



- Too many Usernames and Passwords
- Someone took your desired Username
- User profile is distributed
- Account management is difficult
- Get bored of filling long forms again and again

# How OpenID works??

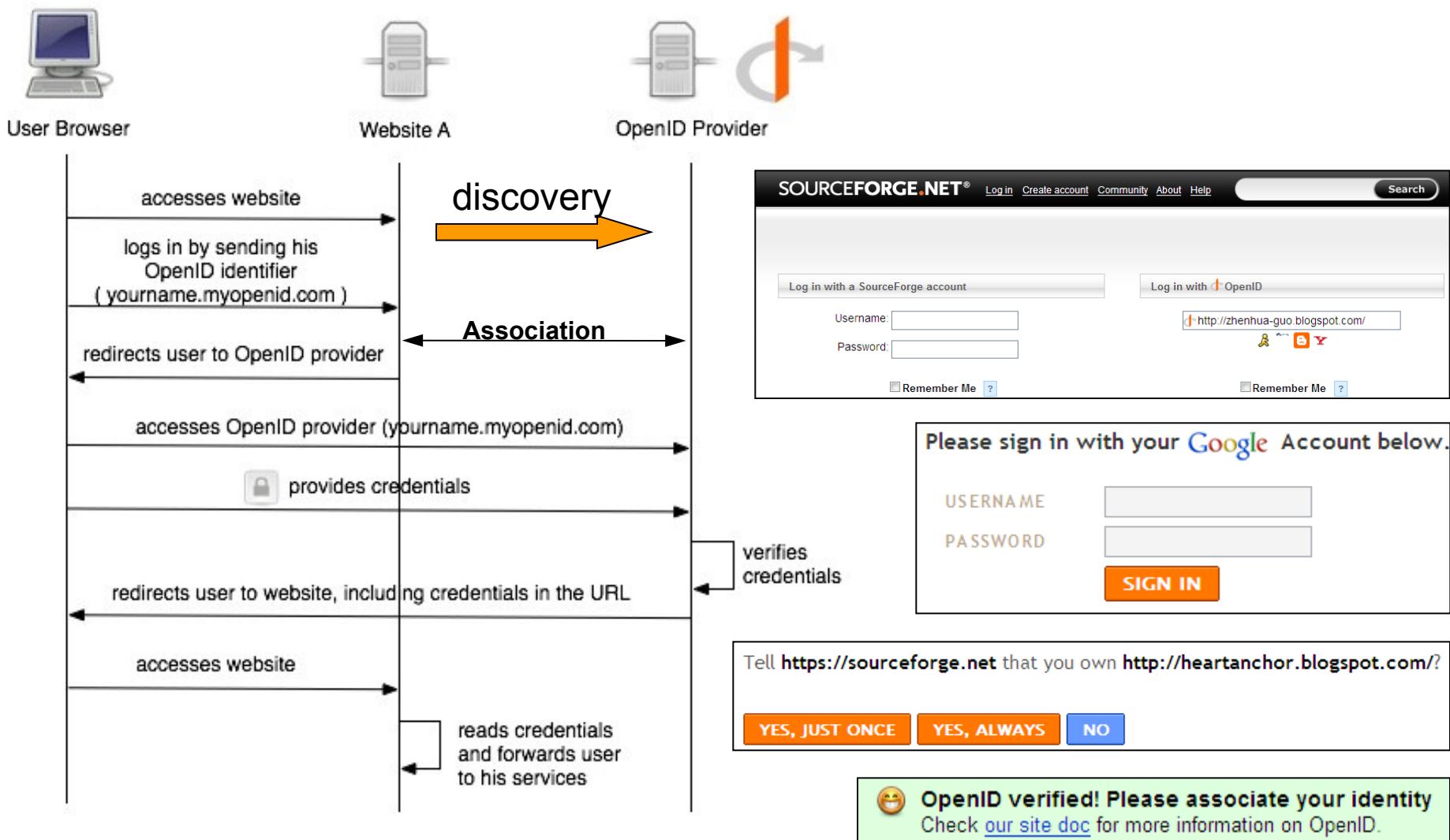


# How openID works?

- Site Fetches the HTML of my openID
- Finds “ openid.server”
- Establishes a shared secret with the provider
- Redirects my browser to the provider where I authenticate and allow the openId login
- Provider redirects my browser back to the site with an openId response.
- Site verifies the signature and logs me in
- OpenID is a decentralized sign-on system for the Web. Not a real single sign-on solution, does not support authorization
- Instead of usernames and passwords, users need to have an account with some identity provider

# OpenID

relying party



# OpenID Support in different Languages

- OpenID Is supported in many programming languages and API's are available
- Java, PHP, Perl C/C++, C#, python ,cold Fusion

# Potential vulnerabilities

- **Phishing.** If the username and password of a user are stolen or phished, then all of the registered sites then become targets.
- A distrusted site redirects you to your trusted provider through a proxy.

## Advantages

- Globally unique & your URL is your Identity
- Few usernames and passwords to remember
- Many OpenID providers like AOL, yahoo, verisignlabs, myOpenID
- Can put OpenID URL on your server also
- Profile data are stored at one place only.
- Control of sharing information.
- Can easily increase business

# OAuth

# Oauth

"An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications."

- oauth.net

- A form of HTTP-based Single Sign On (SSO)
- Similar in spirit to Kerberos, OpenID and SAML
- Primarily focused on Authorization, although it is commonly used for Authentication as well.

## Scope of the OAuth WG

- Oauth 2.0 was published in 2012
- It is not backwards compatible with 1.0.

Documentation here: <https://oauth.net/2/>

There are other documents as individual items

- Providing security related extensions
- User interface considerations
- Token formats
- Token by reference
- Use case descriptions
- Other OAuth profiles

# Benefits

- Authorization and Authentication provided by third party Service Provider
- Application developers can focus on building an app, not an authentication framework
- Username and password are not processed by application
- User identification is collected by service provider
- Improves security since application doesn't need to collect or store the credentials (eg. no clear-text passwords hanging around)

# Benefits

- Centralized management of user accounts
  - Users don't need to create separate account for each application/service
  - Fewer identities & passwords to remember
- Authorization mechanism allows restricting permissions granted to applications
  - User controls whether to grant access or not

# Benefits

- Access may be revoked by user via Service
- Provider account management UI
  - Unified management
  - No need to change password to revoke access
  - User is free to change their password without affecting access granted to application(s)

# Versions of OAuth

- OAuth 1 (RFC 5849)
  - Auth request may be unencrypted
    - Uses HMAC signature to verify source and integrity of request
  - Susceptible to authentication failures if client and server don't agree on argument order and sort differently (signatures won't match)
  - Primary focus is web browser
  - Can provide difficulty authenticating non-browser apps
  - Technically deprecated by OAuth 2

# Versions of OAuth

- OAuth 2 (RFC 6749)
  - Requires SSL for all requests
  - No signatures: No issue with argument ordering
  - Potentially better for larger scale implementations
    - Authentication and authorization can be provided by different servers
    - More suitable for corporate authentication
  - Some feel OAuth2 is not a positive replacement for OAuth1
  - Not backward compatible with OAuth 1
  - Leaves many details up to implementer: Can lead to incompatible, yet compliant, implementations

# Service providers

## Google

- For REST access to Google APIs
- Google+, Drive, AdSense, Analytics, and many more...



- REST and Streaming (real time) APIs



- Graph API for accessing the Facebook social graph.
- Graph API is the basis for most other Facebook APIs.

# Clients of OAuth

- Websites
  - CNN, Washington Post, Gawker, Kickstarter, La Crosse Tribune, etc.
- Mobile apps & games
  - According to Facebook, 81 of the top 100 grossing iOS apps and 62 of the top 100 grossing Android apps use Login with Facebook
- Anything with a "Log in with Facebook/ Google +/Twitter" option

# Oauth 2 General Overview

- Developer registers application with Service Provider
  - Service Provider is: Google, Facebook, Twitter, ...
- Usually requires submission of basic information:
  - Application name
  - Logo
  - Web site or host name
  - Redirect URL

# Oauth 2 General Overview

- Registered applications are issued a Client ID (public) and a Client Secret (private)
- Client Secret is only useful if code and/or binaries of application is not accessible to users
  - Server side web services are a good example
  - Not used by client-side applications (JavaScript, native) since secret would then be available to malicious parties

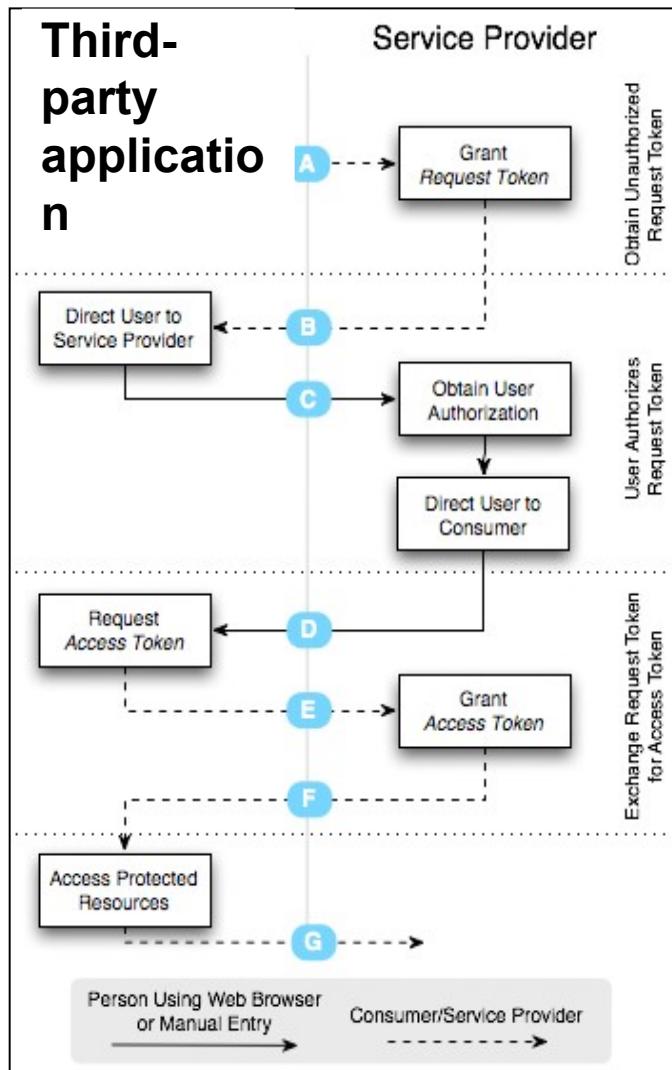
Tokens:

- Request token: used by the consumer to ask the user to authorize access
- Access token: used by the consumer to access the protected resources on behalf of the user

OAuth Authentication is done in three steps:

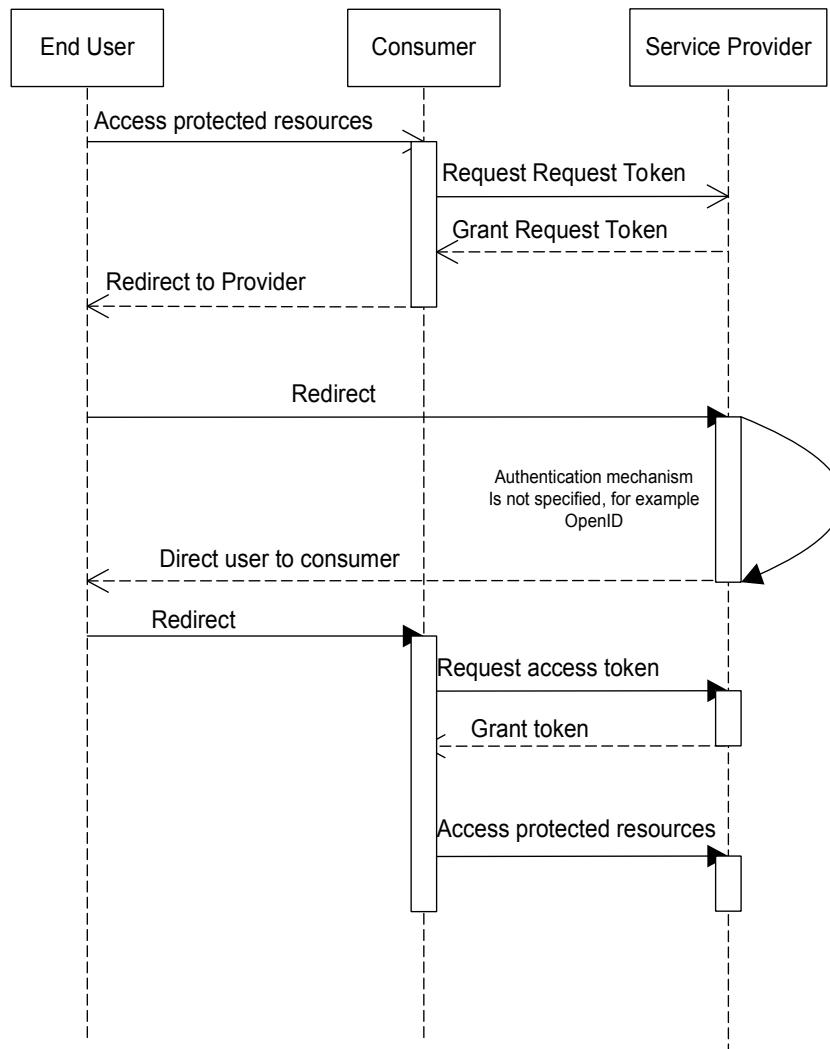
- 1) The Consumer obtains an unauthorized Request Token.
- 2)The User authorizes the Request Token.
- 3)The Consumer exchanges the Request Token for an Access Token.

# OAuth - Flow



Source: <http://oauth.googlecode.com/svn/spec/branches/1.0/drafts/3/spec.html>

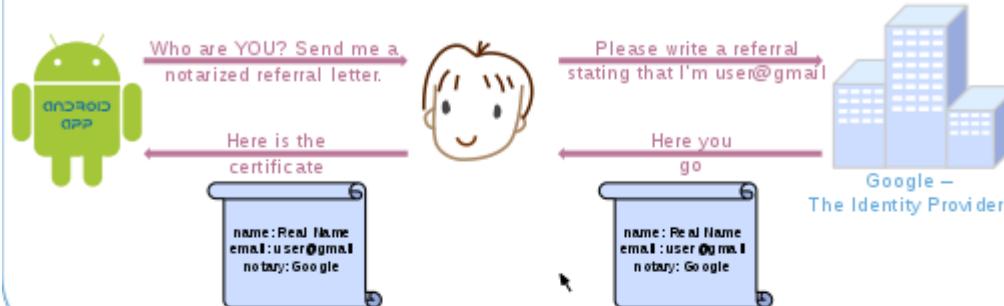
# OAuth - Flow



Source: <http://oauth.googlecode.com/svn/spec/branches/1.0/drafts/3/spec.html>

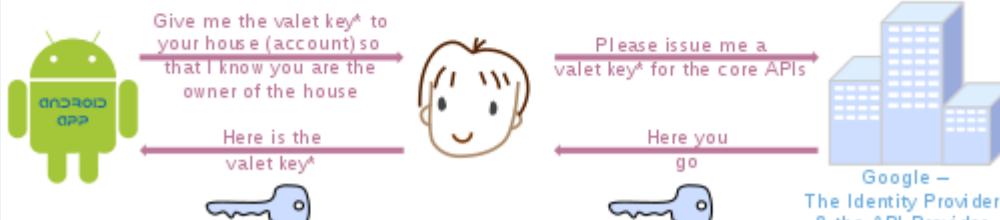
# OpenID vs. OAuth

## OpenID Authentication



vs.

## Pseudo-Authentication using OAuth



\*valet key = limited scope  
OAuth Token

adapted from a drawing by @\_nat\_en

# Potential vulnerabilities

- May be vulnerable to MITM (Man In The Middle) attacks due to reliance on SSL
  - Many mobile apps have been found to be vulnerable to this type of attack
    - Simple(tm) banking app in December, 2012
    - As recently as 2011 (iOS 5), the iOS SSL implementation accepted improperly signed certificates
  - Mitigation:
    - Verify both Issuer & Subject of SSL server certificate issuer must validate against a trusted root CA (either directly or via CA-chain)
    - Subject must validate against site URL

# Potential vulnerabilities

- May be vulnerable to CSRF (Cross Site Request Forgery) attacks
  - Mitigation:
    - Require POST instead of GET to prevent <img> link attacks
    - Submit random state value (CSRF protection) with request and verify on response

```
Response['state'] == Session['state'] && Session['state'] != ""
```

# Potential vulnerabilities

- Vulnerable to XSS (Cross-Site Scripting) attacks
  - Malicious JavaScript could intercept auth tokens
  - Beware of buffer overflow!
    - Attackers may be able to exploit CSRF protection by sending excessively long (4000+ character) state value
    - Browsers/web servers may fail with "HTTP 413 Request URI too long" on redirect, leaving IFrame URL (with valid code!) accessible to malicious JavaScript code

# Potential vulnerabilities

- Reliance on 3rd party service providers for authentication
  - Service may be modified or revoked by Service Provider
  - Application vendor will likely have no control over service outages
  - Example: Feb 7th, 2013 - Facebook auth is down for a short period of time, affecting millions of users on other websites!

# Potential vulnerabilities

Example of an attack:

- In April–May 2017, about one million users of Gmail (less than 0.1% of users as of May 2017) were targeted by an OAuth-based phishing attack, receiving an email purporting to be from a colleague, employer or friend wanting to share a document on Google Docs.
- Those who clicked on the link within the email were directed to sign in and allow a potentially malicious third-party program called "Google Apps" access their "email account, contacts and online documents".
- Within "approximately one hour", the phishing attack was stopped by Google, who advised those who had given "Google Apps" access to their email to revoke such access and change their passwords.

Source: <http://www.bbc.co.uk/news/technology-39845545>

# Social Engineering Attacks

- Users will often blindly approve permission authorization requests
  - Example: Games which have "Log in with Facebook/Google/Twitter"...
  - Some service providers may not clearly indicate which permissions are being granted when presenting confirmation to user
  - Users may not understand the implications of granted permissions
    - Applications could silently monitor user's interactions with OAuth service provider
    - Applications could post content to user's account without user's knowledge

# Social Engineering Attacks

- Once approved, access tokens usually don't expire unless user takes action to revoke access
  - Users may not understand this is even possible or required
  - Access token is not tied to user's browser or device, so may be stored/collected by application vendor
    - Can then be used by vendor even if user stops using application
    - Such use can permit vendor to exceed rate limits on service provider APIs

# Social Engineering Attacks

- Mitigations:
  - User education and awareness
  - Only use websites and applications from reliable sources/developers
  - Read and understand privacy policies
    - If you don't agree with their usage, don't use the app!
  - Periodically review and revoke access to unused applications

# Quiz #1

# Quiz #1

- **Online quiz on Moodle**
- Date: 16.11.2021, at 18:00
- Location: Moodle, MS Teams
- Available time: 30 min
- 10 multiple choice questions

- Example:

A Kerberos client gets a session key from:

- a. the Authentication Server
- b. the Ticket Granting Service
- c. the Application Server

# The End